Learning Human Proxy Functions to Optimize Machine Learning Systems for Sociotechnical Context

HARVARD UNIVERSITY

Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

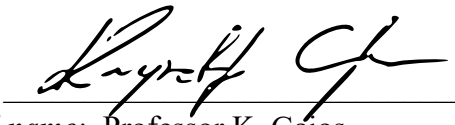
Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Learning Human Proxy Functions to Optimize Machine Learning Systems for Sociotechnical Context"

presented by: Isaac L. Lage

Signature 

Typed name: Professor F. Doshi-Velez

Signature 

Typed name: Professor K. Gajos

Signature 

Typed name: Professor S. Gershman

April 19, 2023

Learning Human Proxy Functions to Optimize Machine Learning Systems for Sociotechnical Context

A DISSERTATION PRESENTED
BY
ISAAC L. LAGE
TO
THE DEPARTMENT OF SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
APRIL 2023

Thesis advisor: Professor Finale Doshi-Velez Isaac L. Lage

Learning Human Proxy Functions to Optimize Machine Learning Systems for Sociotechnical Context

Abstract

Machine learning (ML) systems are increasingly becoming integrated components of human decision making, but many models are developed without considering the sociotechnical context in which they will operate–i.e. how will they be used by humans to solve specific tasks. This mismatch between the development procedure that does not explicitly consider human context, and these models' eventual use by humans, often causes their performance to fall short of expectations (e.g. [21,48]). However optimizing a system to perform well in a specific sociotechnical context requires formalizing the human part of that context in ways that permit optimization. One common approach to this problem is to mathematically specific a *proxy* function for the desired human property that can be easily optimized and evaluated at scale. An intermediary approach between requiring the user to specify their human property at each point where it is needed, and approximating this property by hand-specifying a proxy function is to *learn* a proxy function based on querying a user to evaluate the implicit human function defined by the property at a small number of key points. In this thesis, we formalize an optimization framework that describes this category of approaches; lay out human and machine related desiderata to consider when formalizing a particular instance of a problem under this framework; and present and discuss 3 methods that instantiate this framework to solve different problems in human-ML collaboration.

Contents

Listing of figures

For Misha. Thank you for being there through all of it.

Acknowledgments

I first want to thank my advisor Finale. I am very grateful for the intellectual and emotional support she gave me at each step of the way in this PhD Process. This includes her patience when timelines didn't go as planned, scientific advice about how to approach research problems, and life advice about how to pursue meaningful work and live a life you enjoy. I am also incredibly grateful to her for believing in my ideas, allowing me to pursue them, and encouraging me at times when I lost faith.

Next, I want to express my gratitude to my committee members, Sam and Krzysztof for their feedback on this work, and their mentorship throughout my PhD. Both of them welcomed me into their lab events such as reading groups, group meetings and research conversations that exposed me to different ways of thinking and pushed me to consider other perspectives. I am incredibly grateful for this.

I also want to thank my collaborators from this work. Thank you to Andrew Ross, Been Kim and Sam Gershman who made major contributions to Chapter 3. Thank you to Sonali Parbhoo who made major contributions to Chapter 5. Special thanks to Sonali for her encouragement and for always believing in this work even when I didn't.

I am also incredibly grateful to all of the people who have supported my career both before and during this PhD. Thank you to David Sontag for his research mentorship and encouragement early on in my career that helped me to develop the skills and interests that led me here. Thank you to Weiwei Pan for all of her incredible research and professional advice, her generosity with her time,

and her friendship. Thank you to Ofra Amir for her research collaboration and advice about how to navigate cross-disciplinary research interests. Thank you to Ece Kamar, Besmira Nushi and Eric Horvitz for their research mentorship and their understanding during all the challenges that came along with a COVID internship.

Thank you also to all those who supported my teaching development. Thank you to Mike Hughes for the opportunity to gain valuable teaching practice, to John Girash for his support of my teaching interests, and to everyone at the Bok center for their teaching mentorship.

Thank you as well to everyone in DtAK, both past and current, for all of the camaraderie that got me through this PhD, from game nights, to pranks, to unicorn memorabilia, all of which have made this journey so much brighter and less lonely. Special thanks to Jiayu, Yaniv and Beau for their support throughout all the big milestones.

On a personal note, thank you to my family for all of their support that led me to this point. Thank you to my parents and sibling for their encouragement and for listening to me and supporting me.

Most importantly, thank you to my partner Misha who has been with me throughout this whole process. I wouldn't have been able to do any of this–this PhD or living this life I'm living–without all of their support and for that I'm so, so grateful.

1

Introduction

Machine learning (ML) systems are increasingly becoming integrated components of human decision making, including in the domains of healthcare and self-driving cars. But many models are developed without considering the sociotechnical context in which they will operate–i.e. how will they be used by humans to solve specific tasks. This mis-match between the development procedure that does not explicitly consider human context, and these models' eventual use by humans, often causes their performance to fall short of expectations (e.g. [21,48]). Explicitly optimizing these systems

to perform well in these sociotechnical contexts may help to address this mis-match and increase their performance.

However optimizing a system to perform well in a specific sociotechnical context requires formalizing the human part of that context in ways that permit optimization. For example, if the user of a machine learning system requires that system to be "interpretable," they may be able to recognize whether a specific instantiation of a machine learning system has that property. However turning this property into something that can be used to optimize a machine learning model is not trivial. For example, in order to use gradient descent to optimize this property, it would need to be specified as a differentiable function that can be evaluated for every instance in the training set at every iteration of gradient descent. Having a user directly specify this human property would be challenging because of both the requirement for differentiability, and the scale at which the property must be evaluated.

One common approach to this problem is to mathematically specific a *proxy* function for the desired human property that can be easily optimized and evaluated at scale. Examples include neural network sparsity [24,72], decision tree shallowness [20], integer regression coefficients [90], and decision rule overlap [39] among others. Under this approach, the formalization of this "I'll know it when I see it" type of interpretability would involve specifying a mathematical function that captures this loosely defined property as well as possible. For example, it could be (and has been [43]) formalized as the feature sparsity of a logistic regression model. In that case, "interpretability" can be optimized for by adding an l-1 penalty term to the standard logistic regression loss. Formalizing human properties in this way allows them to be straightforwardly integrated into standard approaches to training ML models.

However, this approach to formalizing human properties works best when it is possible to easily identify a function with "nice" properties (e.g. differentiability) that is a good match for the function implictly defined by the human property. This is often not the case we are operating in, and as

such, this strategy may result in optimizing effectively for properties that poorly capture the human context in which these models will be used (e.g. optimizing models to make good predictions when clinicians want to use them to communicate with patients[28]). In our running example, formalizing "interpretability" as "feature sparsity" will cause the optimization to procedure to ignore other potentially impactful aspects of a good explanation such as causality[52].

An intermediary approach between requiring the user to specify their human property at each point where it is needed, and approximating this property by hand-specifying a proxy function is to *learn* a proxy function based on querying a user to evaluate the implicit human function defined by the property at a small number of key points. This leverages the ML toolkit to generate a *learned proxy function* that can be incorporated into existing machine learning frameworks analogously to the hand-specified proxy functions discussed above, but that also more effectively captures the intended human property.

In this thesis, we consider a collection of methods that approach optimizing machine learning systems to operate in sociotechnical contexts by *learning* mathematical functions of important human properties from well-chosen queries to the user.. Approaching the problem in this way allows us to manage the tradeoff between querying the user too many times to evaluate the human property, and formalizing a human property fully by hand in a process that may miss important aspects of the property. **Our first contribution is to formally define this framework** in Section 2.1.

However many human properties exist, spanning a range of types of human inputs to ML systems, that can formalized under this framework. The methods we discuss in this thesis span the interpretability of a specified model (Chapter 3); the intuitiveness of associations between features and concepts (Chapter 4); and human context that augments each test instance (Chapter 5), but there are many others, some of which may lead to the development of effective methods and some of which may not. Properties may not be well suited to this framework for a variety of reasons: they

may be difficult to people to accurately externalize; they may be difficult to model using the available feature space; and they may not help us solve the desired task. Because of this, choosing a *good* human property to formalize so as to achieve the desired goal is challenging.

How can we approach the problem of finding human properties that *are* useful? **Our second contribution is to characterize a set of "checks" for a given human property to determine how well suited it is to optimizing for sociotechnical context using our framework.** These consist of two parallel sets of human-centric and machine-centric checks that allow us to evaluate how well suited this property is to both solving the original problem, and to the style of method that this framework encompasses. We describe these checks in more detail in Section 2.2.

These checks allow us to evaluate the suitability of a human property for use under this framework, however running these checks is generally expensive as most require either a user study, or the development of a method instantiating this framework for the particular human property. While these steps are necessary to fully evaluate the quality of a proposed method, we would like to make informed choices about methods and corresponding human properties to evaluate that are likely to perform well. **Our third contribution is to provide a "template" for choosing promising properties to develop methods around based on the idea of performing these checks in phases using human proxy functions.**. We describe this template in Section 2.3.2.

To ground this discussion, **our fourth contribution is to describe 3 methods that instantiate this framework for a particular choice of human property, as well as to outline how these checks and the template are used to arrive at and evaluate these methods in iterative steps.**. In each of these methods chapters, we first describe and evaluate the system, then conclude with a short section describing how the method and its evaluation fit into our framework.

The first method focuses on a missing human "interpretability" property that we formalize as how well a user can use a given model to complete a specific task. The method works by simultaneously optimizing the model parameters to be predictive, and learning a proxy function for in-

terpretability based on asking the user to complete the specified task for a small number of model parameters. The second method focuses on the missing human "intuitiveness" property in the context of concept-based models, where concepts must be both predictive and meaningfully defined to the user to be useful for downstream decision-making tasks. In this method, we ask the user directly for their evaluations of the intuitiveness function for carefully chosen pairs of concepts an features. In the third method, we focus on the missing human property of "human features" that are not available to the machine learning model at test time, but that can be elicited from the user for a small number of features in the human feature set. The method works by carefully choosing the features for each instance to elicit from the user based on a model of the human features, and a measure of how knowing their value will impact the model's predictions.

The remainder of this thesis is structured as follows. Chapter 2 describes our proposed framework; set of human and machine "checks" to evaluate a method under our framework; and our "template" to guide development of new methods under this framework. Chapters 3, 4, 5 describe three distinct methods that instantiate this framework, using different types of human input to solve different tasks. Finally, Chapter 6 summarizes our findings, and describes avenues for relevant future work.

2

A Framework for Designing Methods with Learned Human Proxy Functions

In this chapter, we: 1) Formalize the general optimization framework we are working under; 2) Lay out human and machine related desiderata to consider when formalizing a particular instance of a problem under this framework; 3) Describe an approach that can be used as a "template" to develop methods under this framework that work well for both human and machine desiderata.

2.1 Framework Description

The framework we outline assumes the existence of a "human-property" that is needed for some aspect of the machine learning process to operate as desired. These properties can span a range of aspects of model training and deployment, however in all cases, we assume that querying a human for the relevant property every time it is needed during the optimization is cost prohibitive. Instead, we query one or multiple humans to evaluate the property at several key points, then use those "labels" for the human function to fit a *learned proxy function* to approximate the human property. We first describe how we define the implict human function, then how we approximate it, and finally how we optimize for the set of queries.

2.1.1 Defining the Human Function

We assume a generic prediction problem with a dataset D_M consisting of observations X and labels y. I.e. $D_M = \{X, y\}$. These are the standard inputs to a supervised machine learning problem, and our goal is the same–to solve this prediction problem well– i.e. to maximize $\texttt{objective}(y, \hat{y})$ for some predicted labels $\hat{y}$ generated by a machine learning system.

As this is a standard machine learning problem, we can solve it using standard methods to generate a "machine-only" solution that we denote $\hat{y_M} = M(X)$. However, the key novelty in this framework comes from the existence of an auxiliary human-property that can be leveraged by a machine learning system to increase performance on the task. We define an auxiliary component of the dataset, D_H, that consists of set of features, φ and an unobserved set of human function outputs h that implicitly define the human property. I.e. $D_H = \{\varphi, h\}$. Importantly, the function space of this human function is not required to be the same as the function space of the original prediction problem.

The full dataset, D, then consists of $D = \{D_M, D_H\} = \{X, y, \varphi, h\}$. However only $\{X, y, \varphi\}$

are observed–h can only be obtained by asking a user to evaluate their implicit human property at a specific instance φ_i, and each evaluation comes at a cost c as it requires human cognitive effort. Based on this, we can define a "Machine+Human" solution that leverages the human property to better solve the original task: $\hat{y_{M+H}} = M + H(X, h)$. It is uncomputable without observing h, however we define it so as to discuss approximations in later sections.

Finally, we can solve the prediction problem with a "human-only" solution that we denote $\hat{y_H} = H(X, h)$, which consists of a user solving the problem directly without ML assistance.

2.1.2 Approximating the Human Function

As h is unobserved, it cannot be used by a machine learning system without first observing it. While observing h at all points where it is needed is not feasible, we can observe it in key instances by asking a user to give us h_i at a specific point i. We can then approximate the full set of outputs h as $\hat{h} \approx g_q(\varphi)$ by learning it from those key observations, and using the learned proxy $\hat{h}$ where it is needed during optimization and use of the system. Formally, g is an auxiliary machine learning model that can be evaluated across the function space at low cost. The dependence of g on q denotes its dependence on a small set of queries of the true value of h made explicit by the subscript q.

Based on this approximation of h, we can compute an approximate solution of the $M + H$ type as $M + H(X, h) \approx M + H(X, g_q(\varphi))$.

2.1.3 Selecting the Query Set

However as the explicit dependence of the approximation $g_q(\varphi))$ on the query set q denotes, different sets of queries will produce approximate solutions of different quality. As such, the ability to produce a good query set computationally, which we denote as q_M, is at the core of the methods we study under this framework.

While each of the methods we describe solve for this query set in different ways, conceptually, the objective they all seek to optimize is:

$$q^*_M = \operatorname*{argmin}_q \texttt{objective}(y, M + H(X, h)) - \texttt{objective}(y, M + H(X, g_q(\varphi))) \tag{2.1}$$

In other words, we aim to computationally find the set of queries of human function values that allows us to learn an approximation of the human function that solves the original task as well as possible.

We assume that using the true function h is an upper bound on how well we can do–i.e. $\texttt{objective}(y, M + H(X, h)) \geq \texttt{objective}(y, M + H(X, g_q(\varphi))$.

Since we assume $\texttt{objective}(M + H(X, y, h))$ is uncomputable as the full h is too expensive to fully query, we can instead optimize:

$$q^*_M = \arg\max_q \texttt{objective}(M + H(X, y, g_q(\varphi))) \tag{2.2}$$

2.2 Sanity Checks for Methods under this Framework

While this framework spans a large range of possible methods based on different choices of the human property to model, not all of these are equally well-suited to solving the original problem. We specify two key sets of human and machine questions to use when evaluating the particular instantiation of the human property h. These can be used to guide the development and evaluation of new methods. We then describe how these questions can be applied across 3 stages of method development. Finally, we describe a two-phase approach to using these questions during method development that addresses the cost of answering each of these questions, and the dependence between the two sets of questions.

Key Performance Checks

Two parallel sets of human and machine questions or "checks" can be used to evaluate the quality of a particular method instantiating this framework based on a specific human property.

Human Checks

- **HQ1: How much effort is it to give this type of information?**. I.e., how large is c for a particular method? This framework is only necessary if c has an intermediate value where it is feasible to evaluate h at some, but not all, inputs.
- **HQ2: How accurately can users give this type of information?** The more variance there is in the outputs h, the more costly evaluations we will need to do to approximate h well at a particular input.
- **HQ3: How well can the user figure out what stuff to give feedback on?** I.e., how good is $\texttt{objective}(y, \mathcal{M} + H(X, g_{q_H}(\varphi)))$? If the user can select a set of inputs at which to query h that let us approximate it well, the computational methods to choose $q_{\mathcal{M}}$ developed under this framework are not necessary.
- **HQ4: Can the user just solve the problem without the machine assistance?** I.e. how good is $\texttt{objective}(H(X, \varphi, h))$? If the user can solve the original task well, we may not gain utility from developing a method that leverages human and ML insights.

Machine Checks

- **MQ1: How informative is h?** I.e. how good is $\texttt{objective}(y, \mathcal{M} + H(X, h))$? If the human property does not have much utility for solving the original task, it may not be worthwhile to model.

- **MQ2: How well can we computationally model this information?** I.e. how good is the approximation $g_q(\varphi) \approx h$? If we cannot model h well using φ, or it takes too many queries evaluating h to learn it well, the approximation may be too poor to be useful.
- **MQ3: How well can the model figure out what stuff for the user to give feedback on?** I.e. how good is $\texttt{objective}(y, M + H(X, g_{q_M}(\varphi)))$? If we can computationally choose a query set that helps us approximate h well enough to solve the original task, this method may be useful.
- **MQ4: How well can the model solve the problem without the human feedback?** I.e. how good is $\texttt{objective}(y, M(X))$? If the ML system is solving the task well without human input, incorporating h may not be necessary.

2.3 A "Template for Developing Methods Under this Framework

2.3.1 Stages of Method Development

These questions or "checks" can be answered during 3 stages of method development: Defining a prediction problem where human and ML insights are complementary; defining a useful h, and developing a method that optimizes a useful query set q.

Defining a prediction problem where ML and human insights are complementary. . The first stage of developing a method under this framework is to validate whether the task of interest actually requires combining human and ML insights to get a good solution. In many cases, either an expert user (HQ4) or an ML system (MQ4) alone may be able to proficiently solve a task and we may not gain much from formally combining their insights. Importantly, neither of these questions are explicitly dependent on the quality of the proposed method (either a specific h or q),

so they can be evaluated before method development to test the relevance of this style of approach to a general problem setting.

Defining a useful human property h. . The second stage is to clearly articulate what the missing human property is that would allow us to improve the performance of the ML system. This involves both determining where in the process of training and deploying the ML system human insight will be impactful, and then clearly defining a task or question that can be posed to the user to implicitly define the human functions h. The human questions related to defining h address the cost (HQ1) and quality (HQ2) of the human responses when queried for the human property. The machine questions related to h address how well knowing h helps the system solve the original task (MQ1), and how well we are able to computationally approximate h given the feature space available to us (MQ2).

Developing a method to generate useful query sets q. . The last stage of development relates to the quality of the proposed method which optimizes a query set q designed to learn a good approximation of h using as little human effort as possible. The machine question related to this (MQ3) asks how well the proposed method can define a set of queries that efficiently produce good approximations of the human function h. The human question related to this (HQ3) asks how well a user is able to define their own analogous query set to compared to the proposed method. The relative performance of these 2 approaches dictates whether it is, in fact, useful to computationally optimize a set of queries to the user to determine h.

2.3.2 Developing with Human Proxies

While there are distinct stages of method development where each of these checks is applicable, there are two additional challenges that influence when and how to evaluate these questions: 1) the

two sets of questions are dependent on each other, and 2) both are costly to evaluate. The human components of these questions generally rely on expensive user studies, while the machine components rely on the development of a reasonable method for approximating h. Many of these criteria are also relative. For example, in HQ3 and MQ3, we are interested in the relative ability of a human to choose an effective set of queries *compared to* a reasonable and implementable ML approach.

As such, one can approach developing methods under this framework by focusing on one set of questions first, and simplifying the other set. However this may result in methods that are heavily optimized for either the machine or human part, and not for the other. In an extreme case, one might end up finding the optimal human property according to the human questions, however it may not be useful when evaluated against the machine questions, or vice versa. What we want instead is a method that works well enough for both sets of properties.

The three methods we discuss in this thesis address this challenge by considering these questions in two phases. The first phase uses hand-defined computational proxies of the human property instead of a ground truth h queried from users, and the second phase consists of running users studies to evaluate the human questions with that ground truth h. This two phase approach lets us thoroughly test methods as we are developing them without worrying about cost, and only run the expensive user study-based checks once we have a reasonable method that performs well on an approximation of the human-focused questions. However in practice, we do not yet have results for the second phase for two of the methods described in this thesis. We discuss this tradeoff in Chapter 6

Phase 1: testing with proxies for the human property. This framework allows us to learn computational proxies of human properties from human data because in many cases, hand-derived proxies may be insufficient for achieving a particular goal, or they may be impossible to generalize to new domains. However, these hand derived proxies can still be a reasonable approximation

of what a human function might look like. We leverage this insight by testing our method against these hand-derived proxies during development to see if we can learn these reasonable approximations of human functions and if they are useful for solving the task.

Phase 2: testing with real human input. While phase 1 allows us to approximate the answers to the questions that require human input during development, we eventually want to test the method against real human data. In phase 2, we refine our answers to questions involving human components with human data acquired through user studies. This two phase approach allows us to run this expensive step of evaluating with users for methods that appear promising in reasonable approximations to the relevant checks based on proxies for human input.

2.3.3 Related Work

The framework we study in this thesis is one way to approach the problem of developing machine learning systems that work well with humans to accomplish tasks. Other approaches to this include hand-defining proxies for human properties; and developing crowdsourcing techniques for more efficiently acquiring large amounts of human feedback. Examples of this first approach include operationalizing interpretability as the depth of a decision tree [20], the number of integer regression coefficients [90], or the amount of overlap between decision rules [39]. These approaches require a proxy that effectively captures the intended human property accurately, which are challenging to define. Examples of the second approach include Chang et al. [9], which presents a crowdsourcing approach to handling disagreement between labelers on ambiguous labels, and and Tran-Thanh et al. [88] that focuses on algorithmically distributing interdependent subtasks among crowdworkers to reduce labeling cost. These approaches focus on distributing the evaluation of complex human functions among many workers in efficient ways. In contrast, the framework we study uses machine learning to extrapolate a complex human function from a few evaluations which keeps the human cost low,

while still learning from human responses.

Several existing approaches can be described under this framework. Active learning approaches (see Settles [79] for an overview), where the goal is to label a small set of instances to learn a highly predictive model can be seen as a case where the labels are the human function, the set of instances to label is the query set, and the goal is to label the held-out test points as accurately as possible. Approaches for learning human kernels through comparisons like Tamuz et al. [83] can be viewed through this lens where the goal is to learn how humans view things as similar, the similarity evaluations of pairs of inputs are the function output, and choice of which pairs to evaluate creates the query set. While the methods we study are closely related to these, the framework we propose provides structure for defining new types of human functions that appear at various places in the ML system–beyond labels or instance similarity; designing methods around them; and evaluating whether they are appropriate for developing methods that for human-ML collaboration.

Other frameworks for thinking about how to develop ML systems that work well with humans to accomplish a task exist, particularly in the space of interpretability. Notably, Doshi-Velez & Kim [18] present a task focused view of interpretability where it must be evaluated through the performance of users on some defined task using the model. They include 3 types of user evaluation: proxy evaluation, user evaluation with proxy tasks, and user evaluation with real tasks. Chen et al. [11] proposes a step in the evaluation of explanations that relies on evaluation with proxy users. This is similar to the approach that we discuss in the context of human functions specifically tied to interpretability. While we follow a similar approach to these two works in the template recommendation, we focus more broadly on how to develop systems with learned human proxy functions for a wide variety of human-ML collaborations.

2.4 Conclusion and Future Work

In this thesis, we develop a framework for optimizing ML systems to operate in specific sociotechnical contexts based on learning useful proxy functions of important human properties. Our contributions include a formal description of the framework, a set of guidelines for developing methods under this framework, and three case studies of methods that instantiate this framework for different choices of human properties. Under this framework, we can develop methods for optimizing ML systems to operate within specific sociotechnical contexts.

However, there are many avenues for future work in this direction. We focus on two main aspects in this discussion: 1) broadening the set of considerations in the "checks" we propose; 2) other "template" approaches for balancing the cost of evaluating the "checks".

Broadening criteria for success. The checks we provide to evaluate methods focus on performance, but there are other important aspects of decision-making like fairness and human agency. For example, in Chapter 3, the human function evaluation requires users to answer questions that are tedious and not transparently related to the model parameters limiting the agency the user has in this process. In contrast, in Chapter 4, the human function evaluation allows users to give direct feedback on the model parameters in a transparent learning process. While this gives the user more agency over the optimization, this is not formalized in the "checks" we propose.

Future work could expand the checks to include other dimensions of human-ML interactions when considering the success of a particular problem formulation. This could help prioritize the development of methods that not only produce accurate decisions, but also consider qualitative aspects of how users interact with the system.

Considering other template approaches. The "template" we describe provides one way to approach the two problems of the expense of evaluating many of the checks, and the dependence

of the human and machine checks. However there are other possible approaches which may have distinct advantages and disadvantages. Our approach focuses on spending the method development cost before the user evaluation cost by developing methods with human proxies. However this has the disadvantage that one could develop methods that work for proxy human functions, but not the real thing. In Chapter 3, we developed a method without considering the noisiness of the human function. When we evaluated the method with users, we found that the variance of the function estimates were higher than expected because of this. While the method works anyways, and would simply require the collection of more data to reduce the variance, this may not always be the case.

One could alternatively approach the problem with the opposite strategy by first running user studies to evaluate the human properties, then using those results in development. In this case, it would be useful to have a way of guessing whether a specific human property would lend itself to method development in an analogous way to how we use human proxies to guess whether a method will work with human input. Which of these approaches is more appropriate for developing methods under this framework, and in which contexts, is an open question.

In summary, methods that learn an approximation to important, missing human properties can be used to optimize ML systems to operate in sociotechnical contexts without the cost of constantly querying users for the property. In at least some contexts, they also improve over hand-specified proxies that are not learned using human-property data. We present a framework for formalizing methods of this type, and guidelines for developing new methods under this framework.

3

Method 1: Optimizing for the Human Interpretability Function

In this chapter, we use our framework to solve the problem of optimizing ML models to be interpretable, which is challenging because interpretability is a fundamentally human function.

3.1 Introduction

Interpretability is an important property that can help people discover confounders in their training data, and dangerous associations or new scientific insights learned by their models[6,20,43]. But interpretability depends on both the subjective experience of human users and the downstream application, which makes it difficult to incorporate into computational learning methods.

A variety of approaches to producing interpretable models exist, including limiting models to certain function classes that have been *defined* as inherently interpretable (e.g. decision trees[20], generalized additive models[6], decision sets[39]), and optimizing models in more expressive function classes with an interpretability term in the loss function (e.g. sparsity in linear models[85], montone functions[2], ability to be explained by a decision tree[96]). We focus on the second approach as it allows us to explore more expressive function classes, however it requires us to define an interpretability term that can be used in a loss function to train machine learning models.

While the examples we cite are able to do this by optimizing for properties like sparsity, monotonicity and the ability to be explained by decision trees, these properties do not necessarily capture interpretability, as it is a fundamentally human property tied to the ability of a user to perform a task with a model[18]. We approach this by *learning* a human interpretability function over the class of models based on a few well chosen evaluations of the interpretability function.

As evaluating the human interpretability function for a specific model has a high cost—requiring a user study—we develop a cost-effective approach that initially identifies predictive models, then uses model-based optimization to identify an approximate Bayesian MAP solution from that set with few queries to the human interpretability function.

Under our framework, the human interpretability function gives us h, its function space is the space of models M, with feature space φ based on the model's explanation coefficients; and the objective is to maximize the posterior where the interpretability function defines the prior, and the

likelihood is a function of the model's predictiveness. We define this in more detail in Section 3.3. We choose the query set q of models at which to obtain h using model-based optimization, and extrapolate the interpretability function to unqueried models using a Gaussian process.

In our results, we find that different proxies for interpretability prefer different models, and that our approach can optimize all of these proxies. Our human subjects results suggest that we can optimize for human-interpretability preferences.

3.2 Related Work

Learning interpretable models with proxies Many approaches to learning interpretable models optimize proxies that can be computed directly from the model. Examples include decision tree depth [20], number of integer regression coefficients [90], amount of overlap between decision rules [39], and different kinds of sparsity penalties in neural networks [24,72]. In some cases, optimizing a proxy can be viewed as MAP estimation under an interpretability-encouraging prior [85,4]. These proxy-based approaches assume that it is possible to formulate a notion of interpretability that is a computational property of the model, and that we know a priori what that property is. Lavrac [41] shows a case where doctors prefer longer decision trees over shorter ones, which suggests that these proxies do not fully capture what it means for a model to be interpretable in all contexts. Through our approach, we place an interpretability-encouraging prior on arbitrary classes of models that depends directly on human preferences.

Learning from human feedback Since interpretability is difficult to quantify mathematically, Doshi-Velez & Kim [18] argue that evaluating it well requires a user study. Many works in interpretable machine learning have user studies: some advance the science of interpretability by testing the effect of explanation factors on human performance on interpretability-related tasks [61,54] while others compare the interpretability of two classes of models through A/B tests [39,32]. More broadly,

there exist many studies about situations in which human preferences are hard to articulate as a computational property and must be learned directly from human data. Examples include kernel learning[83,94], preference based reinforcement learning[95,14] and human based genetic algorithms[36]. Our work resembles human computation algorithms[45] applied to user studies for interpretability as we use the user studies to *optimize* for interpretability instead of just comparing a model to a baseline.

Model-based optimization Many techniques have been developed to efficiently characterize functions in few evaluations when each evaluation is expensive. The field of Bayesian experimental design[8] optimizes which experiments to perform according to a notion of which information matters. In some cases, the intent is to characterize the entire function space completely[98,49], and in other cases, the intent is to find an optimum[81,80]. We are interested in this second case. Snoek et al.[80] optimize the hyperparameters of a neural network in a problem setup similar to ours. For them, evaluating the likelihood is expensive because it requires training a network, while in our case, evaluating the prior is expensive because it requires a user study. We use a similar set of techniques since, in both cases, evaluating the posterior is expensive.

3.3 Framework and Modeling Considerations

In many cases, the optimization of a property can be viewed as placing a prior over models and solving for a MAP solution of the following form:

$$\max_{M \in \mathcal{M}} p(X|M)p(M) \tag{3.1}$$

where $\mathcal{M}$ is a family of models, X is the data, $p(X|M)$ is the likelihood, and $p(M)$ is a prior on the model that encourages it to share some aspect of our inductive biases. Two examples of biases in-

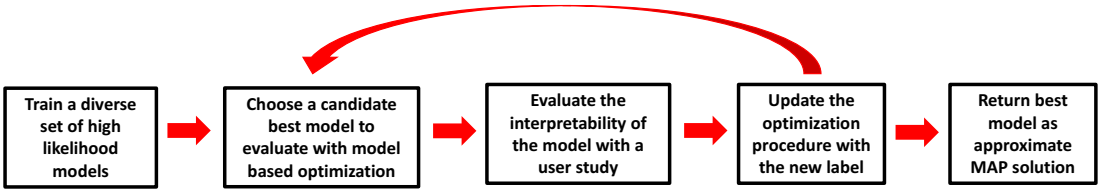


Figure 3.1: High-level overview of the pipeline

clude the interpretation of the L1 penalty on logistic regression as a Laplace prior on the weights and the class of norms described in Bach [4] that induce various kinds of structured sparsity. Generally, if we have a functional form for $p(M)$, we can apply a variety of optimization techniques to find the MAP solution.

Our high-level goal, then, is to find a model M that maximizes $p(M|X) \propto p(X|M)p(M)$ where $p(M)$ is a measure of human interpretability. We assume that computation is relatively inexpensive, and thus computing and optimizing with respect to the likelihood $p(X|M)$ is significantly less expensive than evaluating the prior $p(M)$, which requires a user study. Our strategy will be to first identify a large, diverse collection of models M with large likelihood $p(X|M)$, that is, models that explain the data well. This task can be completed without user studies. Next, we will search amongst these models to identify those that also have large prior $p(M)$. Specifically, to limit the number of user studies required, we will use a model-based optimization approach [81] to identify which models M to evaluate. Figure 3.1 depicts the steps in the pipeline. Below, we outline how we define the likelihood $p(X|M)$ and the prior $p(M)$; in Section 3.4 we define our process for approximate MAP inference.

3.3.1 Likelihood

In many domains, experts desire a model that achieves some performance threshold (and amongst those, may prefer one that is most interpretable). To model this notion of a performance threshold,

we use the soft insensitive loss function (SILF)-based likelihood [15,51]. The likelihood takes the form of

$$p(X|M) = \frac{1}{Z} e^{(-C \times \texttt{SILF}_{\varepsilon,\beta}(1-\texttt{accuracy}(X,M)))}$$

where accuracy(X,M) is the accuracy of model M on data X and $\texttt{SILF}_{\varepsilon,\beta}(y)$ is given by

$$\texttt{SILF}_{\varepsilon,\beta}(y) = \begin{cases} 0, & 0 \leq y \leq (1-\beta)\varepsilon \\ \frac{(y-(1-\beta)\varepsilon)^2}{4\beta\varepsilon}, & (1-\beta)\varepsilon \leq y \leq (1+\beta)\varepsilon \\ y-\varepsilon, & y \geq (1+\beta)\varepsilon \end{cases}$$

which effectively defines a model as having high likelihood if its accuracy is greater than $1-(1-\beta)\varepsilon$.

In practice, we choose the threshold $1-(1-\beta)\varepsilon$ to be equal to an accuracy threshold placed on the validation performance of our classification tasks, and only consider models that perform above that threshold. (Note that with this formulation, accuracy can be replaced with any domain specific notion of a high-quality model without modifying our approach.)

3.3.2 A Prior for Interpretable Models

Some model classes are generally amenable to human inspection (e.g. decision trees, rule lists, decision sets [20,39]; unlike neural networks), but within those model classes, there likely still exist some models that are easier for humans to utilize than others (e.g. shorter decision trees rather than longer ones [71], or decision sets with fewer overlaps [39]). We want our model prior $p(M)$ to reflect this more nuanced view of interpretability.

We consider a prior of the form:

$$p(M) \propto \int_x \texttt{HIS}(x, M) p(x) dx \tag{3.2}$$

In our experiments, we will define $\texttt{HIS}(x, M)$ (human-interpretability-score) as:

$$\texttt{HIS}(x, M) = \begin{cases} 0, & \texttt{mean-RT}(x, M) > \texttt{max-RT} \\ \texttt{max-RT} - \texttt{mean-RT}(x, M), & \texttt{mean-RT}(x, M) \leq \texttt{max-RT} \end{cases} \tag{3.3}$$

where $\texttt{mean-RT}(x, M)$ (mean response time) measures how long it takes users to predict the label assigned to a data point x by the model M, and `max-RT` is a cap on response time that is set to a large enough value to catch all legitimate points and exclude outliers. The choice of measuring the time it takes to predict the model's label follows Doshi-Velez & Kim [18], which suggests this *simulation* proxy as a measure of interpretability when no downstream task has been defined yet; but any domain-specific task and metric could be substituted into our pipeline including error detection or cooperative decision-making.

3.3.3 A Prior for Arbitrary Models

In the interpretable model case, we can give a human subject a model M and ask them questions about it; in the general case, models may be too complex for this approach to be feasible. In order to determine the interpretability of complex models like neural networks, we follow the approach in Ribeiro et al. [69], and construct a simple *local* model for each point x by sampling perturbations of x and training a simple model to mimic the predictions of M in this local region. We denote this $\texttt{local-proxy}(M, x)$.

We change the prior in Equation 3.2 to reflect that we evaluate the `HIS` with the local proxy rather

than the entire model:

$$p(\mathcal{M}) \propto \int_x \texttt{HIS}(x, \texttt{local-proxy}(\mathcal{M}, x))p(x)dx \tag{3.4}$$

We describe computational considerations for this more complex situation in Section 3.4.

3.4 Inference

Our goal is to find the MAP solution from Equation 3.1. Our overall approach will be to find a collection of models with high likelihood $p(X|\mathcal{M})$ and then perform model-based optimization[81] to identify which priors $p(\mathcal{M})$ to evaluate via user studies. Below, we describe each of the three main aspects of the inference: identifying models with large likelihoods $p(X|\mathcal{M})$, evaluating $p(\mathcal{M})$ via user studies, and using model-based optimization to determine which $p(\mathcal{M})$ to evaluate. The model from our set with the best $p(X|\mathcal{M})p(\mathcal{M})$ is our approximation to the MAP solution.

3.4.1 Identifying models with high likelihood $p(X|\mathcal{M})$

In the model-finding phase, our goal is to create a diverse set of models with large likelihoods $p(X|\mathcal{M})$ in the hopes that some will have large prior value $p(\mathcal{M})$ and thus allow us to identify the approximate MAP solution. For simpler model classes, such as decision trees, we find these solutions via running multiple restarts with different hyperparameter settings and rejecting those that do not meet our accuracy threshold. For neural networks, we jointly optimize a collection of predictive neural networks with different input gradient patterns (as a proxy for creating a diverse collection)[73].

3.4.2 Computing the prior $p(\mathcal{M})$

Human-Interpretable Model Classes. For any model $\mathcal{M}$ and data point x, a user study is required for every evaluation of $\texttt{HIS}(x, \mathcal{M})$. Since it is infeasible to perform a user study for every value of x for even a single model $\mathcal{M}$, we approximate the integral in Equation 3.2 via a collection of samples:

$$\begin{aligned} p(\mathcal{M}) &\propto \int_x \texttt{HIS}(x, \mathcal{M}) p(x) dx \\ &\approx \frac{1}{N} \sum_{x_n \sim p(x)} \texttt{HIS}(x_n, \mathcal{M}) \end{aligned}$$

In practice, we use the empirical distribution over the inputs x as the prior $p(x)$.

Arbitrary Model Classes. If the model $\mathcal{M}$ is not itself human-interpretable, we define $p(\mathcal{M})$ to be the integral over $\texttt{HIS}(x, \texttt{local-proxy}(\mathcal{M}, x))$ where $\texttt{local-proxy}(\mathcal{M}, x)$ locally approximates $\mathcal{M}$ around x (Equation 3.4). As before, evaluating $\texttt{HIS}(x, \texttt{local-proxy}(\mathcal{M}, x))$ requires a user study; however, now we must determine a procedure for generating the local approximations $\texttt{local-proxy}(\mathcal{M}, x)$.

We generate these local approximations via a procedure akin to Ribeiro et al. [69]: for any x, we sample a set of perturbations x' around x, compute the outputs of model $\mathcal{M}$ for each of those x', and then fit a human-interpretable model (e.g. a decision-tree) to those data.

We note that these local models will only be nontrivial if the data point x is in the vicinity of a decision boundary; if not, we will not succeed in fitting a local model. Let $B(\mathcal{M})$ denote the set of inputs x that are near the decision boundary of $\mathcal{M}$. Since we defined HIS to equal max-RT when $\texttt{mean-RT}(x, \mathcal{M})$ is 0 as it does when no local model can be fit (see Equation 3.3), we can compute the integral in Equation 3.4 more intelligently by only seeking user input for samples near the model's

decision boundary:

$$
\begin{aligned}
p(\mathcal{M}) &\propto \int_x \texttt{HIS}(x, \texttt{local-proxy}(\mathcal{M}, x)) p(x) dx \qquad (3.5)\\
&= \left(\int_{x \in B(\mathcal{M})} p(x) dx \right) \cdot \left(\int_{x \in B(\mathcal{M})} \texttt{HIS}(x, \texttt{local-proxy}(\mathcal{M}, x)) \tilde{p}(x) dx \right) \\
&\quad + \left(\int_{x \notin B(\mathcal{M})} p(x) dx \right) \cdot \texttt{max-RT} \\
&\approx \left(\frac{1}{N'} \sum_{x_{n'} \sim p(x)} \mathbb{I}(x \in B(\mathcal{M})) \right) \cdot \left(\frac{1}{N} \sum_{x_n \sim \tilde{p}(x)} \texttt{HIS}(x_n, \mathcal{M}) \right) \\
&\quad + \left(\frac{1}{N'} \sum_{x_{n'} \sim p(x)} \mathbb{I}(x \notin B(\mathcal{M})) \right) \cdot \texttt{max-RT}
\end{aligned}
$$

where $\tilde{p}(x) = p(x) / \int_{x \in B(\mathcal{M})} p(x) dx$. The first term (the volume of $p(x)$ in $B(\mathcal{M})$), and the third term (the volume of $p(x)$ not in $B(\mathcal{M})$) can be approximated without any user studies by attempting to fit local models for each point in x (or a subsample of points). We detail how we fit local explanations and define the boundary in Appendix A.3.

3.4.3 Model-based Optimization of the MAP Objective

The first stage of our optimization procedure gives us a collection of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_K\}$ with high likelihood $p(X|\mathcal{M})$. Our goal is to identify the model $\mathcal{M}_k$ in this set that is the approximate MAP, that is, maximizes $p(X|\mathcal{M})p(\mathcal{M})$, with as few evaluations of $p(\mathcal{M})$ as possible.

Let L be the set of all labeled models $\mathcal{M}$, that is, the set of models for which we have evaluated $p(\mathcal{M})$. We estimate the values (and uncertainties) for the remaining unlabeled models—set U—via a Gaussian Process (GP)[68]. (See Appendix A.1 for details about our model-similarity kernel.) Following Srinivas et al. [81], we use the GP upper confidence bound acquisition function to choose among unlabeled models $\mathcal{M} \in U$ that are likely to have large $p(\mathcal{M})$ (this is equivalent to using the

lower confidence bound to minimize response time):

$$\begin{aligned} a_{LCB}(M; L, \theta) &= \mu(M; L, \theta) - \kappa\sigma(M; L, \theta) \\ M_{\mathsf{next}} &= \operatorname{argmin}_{M \in U} a_{LCB}(M; L, \theta) \end{aligned}$$

where κ is a hyperparameter that can be tuned, θ are parameters of the GP, μ is the GP mean function, and σ is the GP variance. (We find $\kappa = 1$ works well in practice.)

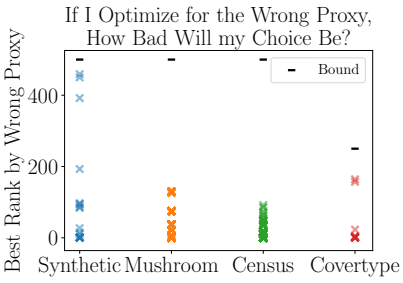
3.5 Experimental Setup

In this section, we provide details for applying our approach to four datasets. Our results are in Section 3.6.

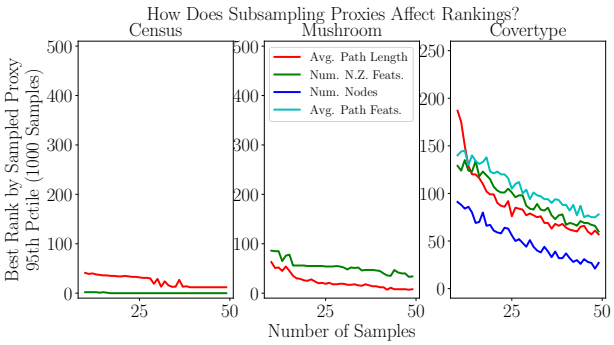
Datasets and Training Details We test our approach on a synthetic dataset as well as the mushroom, census income, and covertype datasets from the UCI database [16]. All features are preprocessed by z-scoring continuous features and one-hot encoding categorical features. We also balance the classes of the first three datasets by subsampling the more common class. (The sizes reported are after class balancing. We do not include a test set because we do not report held-out accuracy.)

- *Synthetic* ($N = 90,000$, $D = 6$, continuous). We build a data set with two noise dimensions, two dimensions that enable a lower-accuracy, interpretable explanation, and two dimensions that enable a higher-accuracy, less interpretable explanation. We use an 80%-20% train-validate split. (See Figure A.1 in the Appendix.)

- *Mushroom* ($N = 8,000$, $D = 22$ categorical with 126 distinct values). The goal is to predict if the mushroom is edible or poisonous. We use an 80%-20% train-validate split.



(a) For each pair of proxies (A, B) for interpretability, we first identify the best model if we only care about proxy A, then compute its rank if we now care about proxy B. This simulates the setting where we optimize for proxy B, but A is the true HIS. This value for each pair of proxies is plotted with an $\times$. The large ranking value indicates that sometimes proxies disagree on which models are good.



(b) Rank of the best model(s) by each proxy across multiple samples of data points ('N.Z.' denotes non-zero and 'feats.' denotes features). This simulates the setting where we compute HIS on a human accessible number of data points. The lines dropping below the high values in Figure 3.2a indicate that computing the right proxy on a human-accessible number of points is better than computing the wrong proxy accurately. This benefit occurs across all datasets and models, but it takes more samples for neural networks on Covertype than the others.

Figure 3.2: Determining interpretability on a few points is better than using the wrong proxy.

- *Census* ($N = 20,000, D = 13$—6 continuous, 7 categorical with 83 distinct values). The goal is to predict if people make more than $\$50,000$/year. We use their 60%-40% train-validate split.

- *Covertype* ($N = 580,000, D = 12$—10 continuous, 2 categorical with 44 distinct values). The goal is to predict tree cover type. We use a 75%-25% train-validate split.

Our experiments include two classes of models: decision trees and neural networks. We train decision trees for the simpler synthetic, mushroom and census datasets and neural networks for the more complex covertype dataset. Details of our model training procedure (that is, identifying models with high predictive accuracy) are in Appendix A.2. The covertype dataset, because it is modeled by a neural network, also needs a strategy for producing local explanations; we describe our parameter choices as well as provide a detailed sensitivity analysis to these choices in Appendix A.3.

Proxies for Interpretability An important question is whether currently used proxies for interpretability, such as sparsity or number of nodes in a path, correspond to some HIS. In the following we will use four different interpretability proxies to demonstrate the ability of our pipeline to identify models that are best under these different proxies, simulating the case where we have a ground truth measure of HIS. We show that (a) different proxies favor different models and (b) how these proxies correspond to the results of our user studies.

The interpretability proxies we will use are: mean path length, mean number of distinct features in a path, number of nodes, and number of nonzero features. The first two are local to a specific input x while the last two are global model properties (although these will be properties of local proxy models for neural networks). These proxies include notions of tree depth[71] and sparsity[43,61]. We compute the proxies based on a sample of 1,000 points from the validation set (the same set of points is used across models).

Human Experiments In our human subjects experiments, we quantify $\texttt{HIS}(x, M)$ for a data point x and a model M as a function of the time it takes a user to simulate the label for x with M. We extend this to the locally interpretable case by simulating the label according to $\texttt{local-proxy}(x, M)$. We refer to the model itself as the explanation in the globally interpretable case, and the local model as the explanation in the locally interpretable case. Our experiments are closely based on those in Narayanan et al. [54]. We provide users with a list of feature values for features used in the explanation and a graphical depiction of the explanation, and ask them to identify the correct prediction. Figure A.3a in Appendix A.4 depicts our interface. These experiments were reviewed and approved by our institution's IRB. Details of the experiments we conducted with machine learning researchers and details and results of a pilot study conducted using Amazon Turk are in Appendix A.4.

Optimizing different automatic proxies results in different models. For each dataset, we run simulations to test what happens when the optimized measure of interpretability does not match the true HIS. We do this by computing the best model by one proxy–our simulated HIS, then identifying what *rank* it would have had among the collection of models if one of the other proxies–our optimized interpretability measure–had been used. A rank of 0 indicates that the model identified as the best by one proxy is the same as the best model for the second proxy; more generally a rank of r indicates that the best model by one proxy is the rth-best model under the second proxy. Figure 3.2a shows that choosing the wrong proxy can seriously mis-rank the true best model. This suggests that it is not a good idea to optimize an arbitrary proxy for interpretability in the hopes that the resulting model will be interpretable according to the truly relevant measure. Figure 3.2a also shows that the synthetic dataset has a very different distribution of proxy mis-rankings than any of the real datasets in our experiments. This suggests that it is hard to design synthetic datasets that capture the relevant notions of interpretability since, by assumption, we do not know what these are.

Computing the right proxy on a small sample of data points is better than computing the wrong proxy. For each dataset, we run simulations to test what happens when we optimize the true HIS computed on only a small sample of points–the size limitation comes from limited human cognitive capacity. As in the previous experiment, we compute the best model by one proxy–our simulated HIS. We then identify what rank it would have had among the collection of models if the same proxy had been computed on a small sample of data points. Figure 3.2 shows that computing the right proxy on a small sample of data points can do better than computing the wrong proxy. This holds across datasets and models. This suggests that it may be better to find interpretable models by asking people to examine the interpretability of a small number of examples—which will result in noisy measurements of the true quantity of interest—rather than by accurately optimizing a proxy

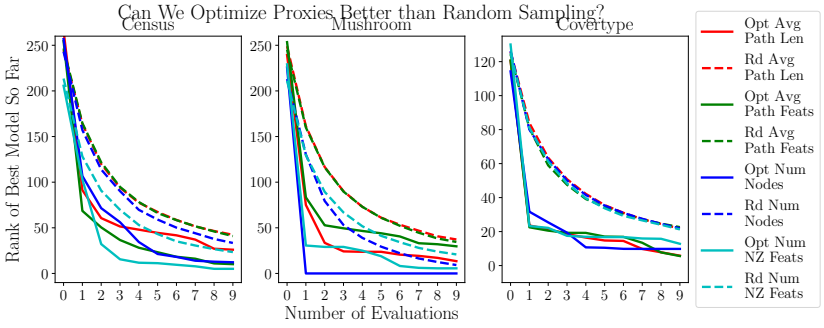
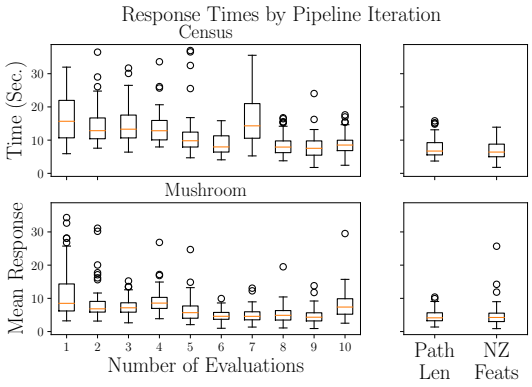


Figure 3.3: We ran random restarts of the pipeline with all datasets and proxies–denoted 'opt' (randomness from choice of start), and compared to randomly sampling the same number of models–denoted 'rd' (we account for models with the same score by computing the lowest rank of any model with that score). 'NZ' denotes non-zero and 'feats' denotes features. The fact that the solid lines stay below the corresponding dotted lines indicates that we do better than random guessing.

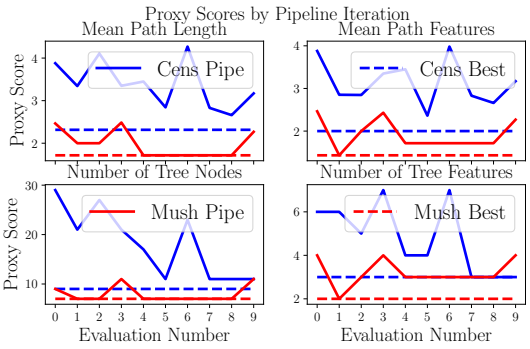
that does not capture the quantity of interest.

Our model-based optimization approach can learn human-interpretable models that correspond to a variety of different proxies on globally and locally interpretable models. We run our pipeline 100 times for 10 iterations with each proxy as the signal (the randomness comes from the choice of starting point), and compare to 1, 000 random draws of 10 models. We account for multiple models with the same score by computing the lowest rank for any model with the same score as the model we sample. Figure 3.3 shows that across all three datasets, and across all four proxies, we do better than randomly sampling models to evaluate.

Our pipeline finds models with lower response times and lower scores across all four proxies when we run it with human feedback. We run our pipeline for 10 iterations on the census and mushrooms datasets with human response time as the signal. We recruited a group of machine learning researchers who took all quizzes in a single run of the pipeline, with models iteratively chosen from our model-based optimization. Figure 3.4a shows the distributions of mean response times decreasing as we evaluate more models. (In Figure A.3b in Appendix A.4 we demonstrate that increases in speed from repeatedly doing the task are small compared to the differences we see in Fig-



(a) We computed response times for each iteration of the pipeline on two datasets. Each data point is the mean response time for a single user. In both experiments, we see the mean response times decrease as we evaluate more models. We reach times comparable to those of the best proxy models. The last 2 models are our baselines ('NZ feats' denotes non-zero features).



(b) We computed the proxy scores for the model evaluated at each iteration of the pipeline. On the mushroom dataset, our approach converges to models with the fewest nodes and shortest paths, and on the census dataset, it converges to models with the fewest features. 'Mush' denotes the mushroom dataset and 'Cens' denotes the census dataset.

Figure 3.4: Human subjects pipeline results show a trend towards interpretability.

ure 3.4a; these are real improvements in response time.)

On different datasets, our pipeline converges to different proxies. In the human subjects experiments above, we tracked the proxy scores of each model we evaluated. Figure 3.4b shows a decrease in proxy scores that corresponds to the decrease in response times in Figure 3.4a (our approach did *not* have access to these proxy scores). On the mushroom dataset, our approach converged to a model with the fewest nodes and the shortest paths, while on the census dataset, it converged to a model with the fewest features. This suggests that, for different datasets, different notions of interpretability are important to users.

3.7 Discussion and Conclusion

We presented an approach to efficiently optimize models for human-interpretability (alongside prediction) by directly including humans in the optimization loop. We were able to do this by de-

veloping a method under our framework where the human interpretability function outputs are h, and we efficiently learned a computational proxy for it that allowed us to optimize models for this property using a Gaussian Process. Upper confidence bound search allowed us to choose an effective query set q where we elicited human evaluations of the interpretability function through user studies.

To evaluate this method, we answered some of the checks by either assumption or design, and some by first using proxy human functions, then running the entire pipeline with user studies to refine the estimates of those questions. We answered HQ4 by assumption, assuming that an ML system is needed to find a predictive model; HQ1-2 through task design, with a user study task that was doable and concretely linked to interpretability as we defined this property through the ability to complete the task; finally we assumed that, due to the lack of transparency in how human inputs influenced the optimization, people would not be able to choose an affective query set. While we were not directly evaluating HQ2 in our human results, we did find that the noise in the human function was higher than expected, suggesting that a more thorough consideration of HQ2 could have been useful.

Our experimental results largely focused on the machine questions, first evaluating them with human proxy functions, then with real human data. MQ1 and MQ4 are closely linked, as we defined the interpretability function as a piece of the objective, so knowing it is directly relevant to solving the goal. We did, however, explore whether the human proxy functions were sufficient for doing this optimization and found that they were not. MQ2 and MQ3 we evaluated through the performance of the pipeline, finding that the human function could be learned sufficiently well to optimize models according to the objective. We showed these results through simulations with proxy functions to simulate h, then ran a human study that provided additional evidence towards answering these questions.

In summary, these experiments allow us to show that, across several datasets, several reasonable

proxies for interpretability identify different models as the most interpretable; all proxies do not lead to the same solution. Our pipeline was able to efficiently identify the model that humans found most expedient for forward simulation. While the human-selected models often corresponded to some known proxy for interpretability, which proxy varied across datasets, suggesting the proxies may be a good starting point but are not the full story when it comes to finding human-interpretable models.

That said, the direct human-in-the-loop optimization has its challenges. In our initial pilot studies with Amazon Mechanical Turk (Appendix A.4), we found that the variance among subjects was simply too large to make the optimization cost-effective (especially with the between-subjects model that makes sense for Amazon Mechanical Turk). In contrast, our smaller but longer within-subjects studies had lower variance with a smaller number of subjects. This observation, and the importance of downstream tasks for defining interpretability suggest that interpretability studies should be conducted with the people who will use the models (who we can expect to be more familiar with the task and more patient).

The many exciting directions for future work include exploring ways to efficiently allocate the human computation to minimize the variance of our estimates $p(M)$ via intelligently choosing which inputs x to evaluate and structuring these long, sequential experiments to be more engaging; and further refining our model kernels to capture more nuanced notions of human-interpretability, particularly across model classes. Optimizing models to be human-interpretable will always require user studies, but with intelligent optimization approaches, we can reduce the number of studies required and thus cost-effectively identify human-interpretable models.

4

Method 2: Optimizing Models for Human-Concept-Alignment

In this chapter, we approach the problem of learning concept-based models–models that can be explained in terms of high-level, human meaningful concepts by formalizing a human-intuitiveness property that allowed us to constrain the set of possible concepts to quickly find intuitive ones.

4.1 Introduction

Concept-based machine learning methods express predictions in terms of high-level concepts derived from raw features instead of in terms of the raw features themselves. In general, the goal is for machine-learned concepts to align with human users' internal concepts (rather than simply being representations that increase the machine's predictive performance). Because the concepts are intuitive to users, they can facilitate interpretation and interaction with models in a way not possible at the level of the high-dimensional raw features. For example, Kim et al. [34] uses concepts to understand biases in neural networks trained on image data, and Koh et al. [35] argues that models with a concept bottleneck layer can be intervened on to make correct predictions even when concept predictions are incorrect.

In this work, we focus on methods to identify concepts from the kinds of high-dimensional, tabular, count data that frequently occur in healthcare records. In these settings, the raw data consist of quite granular codes (e.g. patient has codes for Lorazepam and generalized anxiety), and the clinician's mental model operates at a higher level of patient condition (e.g. patient has anxiety). Methods to turn these raw features into concepts that clinicians can easily reason about, and then explain predictions in terms of these concepts, should be easier to understand and manipulate than those expressed in terms of raw diagnostic codes.

However existing concept-based models have limitations that make them difficult to apply in settings like the clinical one described above. The first major limitation is that these methods define concepts in terms of black-box classifiers, so they may fail to accurately capture the user's mental model of the concept in any of the number of ways that black box models may fail (spurious correlations, lack of fairness, etc.), and without tools to interpret *the concepts*, a user's attempt to interpret the predictive model based on them may also fail. The second major limitation is that these methods rely on access to ground-truth concept labels for at least some fraction of the dataset. However, in

many cases this annotation can require significant user effort. This is exacerbated by the fact that the process of chart review to obtain a patient condition label from health data is much more time-consuming than labeling an image, which is the setting that many previous works have considered.

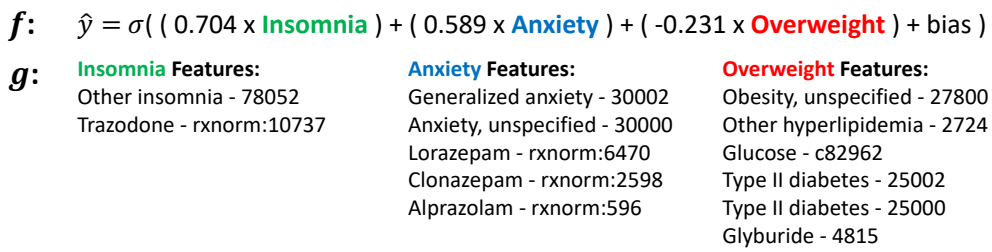


Figure 4.1: An example model of the form used by our method in a clinical domain. The g component is a transparent representation layer that generates intuitive concepts, and the f component is a transparent model learned on top of the representation.

We propose an approach to learning concept-based models that addresses both of these limitations. We learn a concept-based model where the concepts are fully transparent, thus enabling users to validate whether mental and machine models are aligned. Our learning process also incorporates user feedback directly when learning the concept definitions: rather than labeling data, users mark whether specific *feature dimensions* are relevant to a concept. Under our framework, h here is the relevance of a feature to a concept, or the intuitiveness of an association between the two, and its function space is the set of feature-concept pairs.

This choice of h, rather than eliciting concept labels for instances which is the choice of h used in the related work we reference, further enhances human-machine concept alignment by allowing the user more control over the learned concept, and makes efficient use of the user's time, particularly since Raghavan et al. [66] found that labeling a feature takes a fraction of the time of labeling an instance.

Our objective, then, is to maximize the model's predictive performance while adhering to the alignment of the concept with the user's internal concepts operationalized through this choice of

h, and some programatically defined constraints on interpretability. (See Figure 4.1 for examples of concept definitions of the form proposed in our method.) We demonstrate with simulated user feedback on real prediction problems, including one in a clinical domain, that this kind of direct feedback is much more efficient at learning solutions that align with ground truth concept definitions than alternative transparent approaches that rely on labeling instances or other existing interaction mechanisms, while maintaining similar predictive performance.

4.2 Related Work

Transparent Machine Learning. Transparency has been proposed as one instantiation of interpretability corresponding to whether a user can step through a model's computation in a reasonable amount of time[44]. It can help users avoid pitfalls of black box models like unfairness, relying on spurious correlations, and a variety of other errors[74]. Many machine learning models have been proposed to satisfy this criteria including sparse logistic regression (Tibshirani[86], Poursabzi-Sangdeh et al.[62]), decision sets (Lakkaraju et al.[39], Lage et al.[37]) and rule lists (Ustun & Rudin[89], Letham et al.[42]). However these approaches are presented in terms of raw features, assuming they are meaningful. Our work aims to develop a transparent representation that maps uninterpretable features in high-dimensional domains to an intuitive and transparent concept representation on top of which existing transparent machine learning methods can be used.

Uninterpretable Concept-Based Models. Approaches have been proposed to give intuitive meaning to latent spaces of complex (uninterpretable) models. Most closely related to ours, Koh et al.[35] introduces an approach for constructing a bottleneck layer in a neural network that corresponds to a set of concepts that are labeled in the training data. They demonstrate that these concepts can then be manipulated to change and understand the model's output instead of directly manipulating input features. Other related lines of work use semi-supervised approaches where a

user provides some labels for latent dimensions in order to constrain the latent space to align with the user's representation of the problem (Narayanaswamy et al. [55], Hristov et al. [26]), or interpret the latent space of a neural network in terms of intuitive concepts post-hoc by allowing users to specify concepts in terms of examples that train a classifier on a neural network's latent space [33]. In contrast to this, our method learns a model that is fully transparent, not post-hoc, and does not require a dataset of instances labeled with concepts.

Interactive Concept Learning. Approaches to interactively learn a concept-based representation have been proposed, several of which produce interpretable concept-based representations. Amershi et al. [3] introduces "end-user interactive concept learning," through which users can generate labels for relevant concepts in large datasets to train concept classifiers, which can be done with transparent classifiers. Interactive topic models have been proposed to obtain low-dimensional representations aligned with user intuition; these are linear, positive and can be interpreted by the top features for each topic. Mechanisms for interaction include constraints that words should or should not appear in the same topic, [27], and defining a set of "anchor words" to characterize a topic; the latter is considered easier to guide. Lund et al. [47] builds on it to learn predictive topics for downstream prediction tasks. Finally, Parikh & Grauman [57] learns mid-level features for image classification by jointly finding predictive hyper-planes, and learning a model to predict the nameability of those hyper-planes, but this depends heavily on users being able to inspect instances to see how a latent feature varies among them. This works well with images, but it is not clear how this would work with data that is not glaneceable. While these methods allow for interactively learning interpretable concepts, the feedback users provide does not allow them to directly steer the algorithm, making it challenging for users to get the concepts to align with their intuitive representation of the problem.

Interactive Feature Engineering. Additional methods have been proposed to allow users to interactively engineer and refine feature spaces. Cheng & Bernstein [12] and Takahama et al. [82] use crowd-workers to label images with features generated by other crowd-workers, then iteratively refine the feature space for examples that are frequently misclassified. Some active learning methods have been proposed that allow features to give feedback on the relevance of features rather than labeling instances e.g. [66,19]. In user studies, Raghavan et al. [66] and Druck et al. [19] demonstrated that people are able to label features with their relevance to a prediction task, and Raghavan et al. [66] demonstrated that labeling features actually takes much less time than labeling instances. In contrast to our method, these all aim to increase predictive performance of the downstream model with user feedback, rather than to tune the model to be intuitive to the user.

4.3 Interactively Learning Intuitive Concepts

Our goal is to predict label $y \in \{0,1\}$ given input feature vector of count data: $x \in \mathcal{Z}^{+D}$. We assume access to a dataset $\{x_n, y_n\}^N$ and aim to learn a 2-stage prediction function first mapping from input features x to concepts $c \in \{0,1\}^C$–we call this the concept definition, g, and then mapping concepts c to predictions $\hat{y}$–we call this the prediction function, f.

Our goal is to learn an f and g that are as predictive as possible, while learning an f that is interpretable, and a g that is both interpretable and intuitive, in that it closely corresponds to a set of concepts the user is already familiar with. This gives us the following objective function that we solve:

$$\begin{aligned} \arg\max_{f,g} \quad & \texttt{accuracy}(f(g(x)), y) \\ \text{subject to} \quad & f \text{ is interpretable} \\ & g \text{ is interpretable} \\ & g \text{ is intuitive} \end{aligned} \tag{4.1}$$

Below, we instantiate f and g which have functional forms that are transparent by design. Then we introduce an optimization procedure to fit the parameters of f and g so that they have the correct functional form, g is intuitive–i.e. aligned to the user's internal concept definition, and $(f(g(x)))$ is predictive. Section 4.4 describes a core piece of the optimization framework in more detail.

4.3.1 Interpretable Concept Definition $c = g(x)$

The key claim underlying our method is that, if the concept definitions g produce concepts that closely align with the user's internal concepts, then the entire predictive model consisting of the functions f and g can be interpreted by only inspecting f. This reduces the effort required to interpret the model. To accomplish this, we require that concept definition g be interpretable so that the user can validate that it produces concepts that closely match their own. Without this requirement, problems that interpretability aims to solve, such as models learning spurious correlations, can creep into g and influence the interpretation of the entire model.

To ensure that the learned concept definitions g are interpretable, we draw inspiration from the medical literature where conditions are often manually defined from high dimensional record data. One common form is defining a condition (one possible instantiation of a concept) based on a threshold on a sum of counts. This form of concept definition is known to be interpretable to humans since it is the de-facto clinical approach (e.g. Castro et al.[7], Townsend et al.[87], Ritchie et al.[70]).

We formalize a simplified version of this popular form without a count threshold by defining the concept definition g using a binary matrix of parameters, $A \in \{0,1\}^{D \times C}$ where setting $A_{i,j} = 1$ indicates that feature i is associated with concept j. A prediction c_j can then be made for concept j as follows:

$$c_j = 1((A_j x) \geq 1) \tag{4.2}$$

Features i where $A_{i,j} = 1$ form a list of features that are associated with concept c_j. The main goal of our approach will be to learn this set of features.

4.3.2 Interpretable Prediction Function $\hat{y} = f(c)$

Since our goal is to interpret the model in terms of the concepts instead of the raw input features, the prediction function f that depends on the concepts must be interpretable. In this work, we shall use logistic regression, but in general, any differentiable and interpretable model could be used. Let $W \in \mathcal{R}^C$ be the vector of weights and $b \in \mathcal{R}$ the scalar bias. The prediction can be written:

$$\hat{y} = \sigma(W^T c + b) \tag{4.3}$$

The model can then be interpreted in terms of the concept weights, W.

4.3.3 Optimization procedure

Given the forms we have defined for f and g, we must define a procedure for solving our objective in Equation 4.1. The main challenge to our optimization is the last constraint: that g must be intuitive. Since intuitive concepts are a property of the user's understanding of a particular domain rather than a property of the data itself, we must assess this property with the help of the user.

We define an optimization procedure that relies on being able to query the user about whether a feature x_i should be associated with concept c_j. For example, an association between the feature 'generalized anxiety' and the concept 'anxiety' might make sense, whereas an association with 'anxiety' and 'other insomnia' does not—even though it might make the concept more predictive of the patient's psychiatric outcomes. We assume that if the user accepts associating feature x_i with concept c_j for every (i, j) feature-concept association in g, then g is intuitive and satisfies the constraint. I.e. the constraint is satisfied if the user accepts associating x_i with $c_j \; \forall \{(i, j) : A_{i,j} = 1\}$.

To guarantee that we will learn a g that satisfies this property, we first require the user to initialize g with an intuitive association for each concept, grounding the concepts to something the user finds intuitive. Our algorithm then iteratively builds up g by making a series of feature-concept proposals (i^*, j^*) that the user must accept or reject. If the proposal is accepted, the feature-concept association is added to the concept definition, improving the quality of g while guaranteeing that it satisfies the constraint, and if it is rejected, the concept definition does not change so g continues to satisfy the constraint.

We also make efficient use of human feedback by modeling which associations the user has previously accepted to refine future proposals made by the algorithm, and we refit f after every change to g to increase the overall predictiveness of the model. Each of these steps is described below, and the full algorithm is described in Algorithm 1.

INITIALIZATION The user first initializes the binary feature-concept association matrix, A, by specifying exactly one feature they wish to be associated with each concept: $\forall j \sum_{i=0}^{D} A_{(i,j)} = 1$.) In clinical domains like the Psych one we use in our experiments, clinicians already know many of the high-level concepts of interest that will affect the prediction, and coming up with a few examples of features related to these concepts (e.g. generalized anxiety) seems relatively straightforward, but specifying the long tail of relevant features (e.g. alprazolam, lorazepam etc.) can be challenging; this is where our approach is useful.

PROPOSALS The optimization then proceeds by, at each step, proposing a feature-concept pair, (i^*, j^*) from the set of $\{(i, j)\}$ pairs that have not yet been explored by the algorithm. We call the unexplored set of features associated with concept c_j: u_j (for unlabeled), and the previously explored set l_j (for labeled). The user then gives feedback by either accepting the association, thus setting feature-concept matrix $A_{i,j} = 1$, or by rejecting the association, in which case no change is made to

A. l_j is initialized as $\{i : A_{i,j'} = 1\}$ for any concept j', and u_j is initialized as the complement of l_j. This is further described in Section 4.4.

Labels for intuitive associations We additionally store a set of labels indicating where the user has previously accepted or rejected a proposal: `intuit`. These allow us to model which associations are or are not intuitive, and to refine the quality of future proposals. These are initialized so that $\texttt{intuit}_{i,j} = 1$ and $\texttt{intuit}_{i,j' \neq j} = 0$ if $A_{i,j} = 1$ in the concept definitions initialized by the user. We then update these intuitiveness labels setting $\texttt{intuit}_{i^*,j^*} = 1$ if proposal (i^*, j^*) is accepted and $\texttt{intuit}_{i^*,j^*} = 0$ if it is rejected. In the initialization, we assume that features can only be associated with one concept, but we do not make this assumption for our proposals. This reflects the assumption that features used to seed the concepts will be more cleanly aligned with them than features suggested by our algorithm.

Additional details After each accepted change to the feature-concept matrix A, we refit f so that it is optimally predictive given the updated concept definitions. We fix the concept for each proposal, j^*, optimizing only over the feature dimension, i^*. We make a fixed number of proposals to the user for each concept before moving onto the next in order to minimize the mental load placed on the user by continuously switching between concepts. To short-circuit the loop of human feedback when no predictive proposals exist, we add a dummy feature that corresponds to making no change to the concept, and do not request feedback from the user when this is proposed.

Algorithm 1 Our algorithm for interactively optimizing intuitive and interpretable concepts with human feedback. See Section 4.4 for details on how we choose feature i^* when making proposals.

1: Input: x, y, A, k
2: Initialize: f, l, u, `intuit`
3: **for** j^* 1:num-concepts **do**
4: **for** k 1:num-proposals **do**
5: Choose feature i^* to construct proposal: (i^*, j^*)
6: **if** (i^*, j^*) is accepted **then**
7: $\texttt{intuit}_{i^*,j^*} = 1$
8: $A_{i^*,j^*} = 1$
9: Retrain f
10: **else**
11: $\texttt{intuit}_{i^*,j^*} = 0$
12: **end if**
13: $l_{j^*} = l_{j^*} \cup \{i^*\}; u_{j^*} = u_{j^*} \setminus \{i^*\}$
14: **end for**
15: **end for**

4.4 Proposing Predictive, Likely-Intuitive Changes

The key challenge in the optimization process outlined above is how to make proposals that will both increase the predictive performance, and will also likely satisfy the intuitiveness constraint– i.e. how to implement Line 5 in Algorithm 1. If the proposal is not highly predictive, it will not improve the objective much even if the user accepts it (and thus be a waste of the user's time), while

if the proposal is not intuitive, that is, not aligned to the user's mental model, then the user will not accept it and no improvement will occur.

To produce proposals that are predictive and intuitive, we first compute both how predictive a particular feature-concept association is likely to be—we denote this $\texttt{score}^{\texttt{pred}}$—and how likely to user is to accept it—we denote this $\texttt{score}^{\texttt{intuit}}$. Next we combine both these scores—we denote the combiner function $h_k(\texttt{score}^{\texttt{pred}}, \texttt{score}^{\texttt{intuit}})$, to rank potential proposals. k corresponds to a threshold parameter used by the combiner function. Below, we describe the procedures for computing $\texttt{score}^{\texttt{pred}}$ and $\texttt{score}^{\texttt{intuit}}$ before describing how they are combined in h_k.

COMPUTING $\texttt{score}^{\texttt{pred}}$ The goal of $\texttt{score}^{\texttt{pred}}_{i,j}$ is to measure how much the predictive performance of the model will increase if feature i is associated with concept j. We want to use this score to prioritize proposals that will improve the predictive performance as much as possible if the user decides to accept the proposal.

We achieve this by updating the model to its form if the proposal (i, j) were to be accepted by the user, and computing the predictive performance under this update (without retraining f because of the computational cost). We denote this update $\tilde{A}^{i,j}_{i,j}$, and it is identical to A except that $\tilde{A}^{i,j}_{i,j} = 1$. Formally, $\texttt{score}^{\texttt{pred}}_{i,j} = \texttt{accuracy}(y, f(g(x, \tilde{A}^{i,j}))$ for $i \in u_j$, and 0 o.w. While brute force approaches like this can be computationally expensive, in this case, it allows us to compute the $\texttt{score}^{\texttt{pred}}_{i,j}$ quantity in only several seconds (each proposal for the Yelp domain took approximately 2 seconds on a single core of an Intel Core i7 processor). This appears to be within the bounds of reasonable behavior for an interactive system; Lund et al. [47] report that participants were unable to complete a task when updates took more than 10 seconds.

COMPUTING $\texttt{score}^{\texttt{intuit}}$ The goal of $\texttt{score}^{\texttt{intuit}}_{i,j}$ is to measure how likely the user is to accept associating a feature i with a concept j. When this is high, we can propose a feature-concept pair

with confidence, knowing that the user is likely to accept the change. When it is low, we should prioritize other proposals that the user is more likely to accept.

We operationalize $\texttt{score}^{\texttt{intuit}}$ using a Gaussian random field (GRF)[98] to model the user's probability of accepting a proposal. This model assumes that the user is likely to accept associating a feature i with a concept j if the user has previously accepted associating similar features i' with concept j and unlikely to do so if they have previously rejected similar features. This requires defining a notion of similarity between features: we use Jaccard similarity (denoted J) computed over the number of times each features is recorded for each instance (i.e. x^T). Synonymous features are likely to be used somewhat interchangeably throughout a patient's medical history, for example, making this notion of similarity reasonable.

The probability that the user will accept associating feature i with concept j can then be computed based on propagating the `intuit` labels, denoting whether the user has previously accepted or rejected a proposal, through the similarity graph:

$$\texttt{score}_{i,j}^{\texttt{intuit}} = \exp\left(\frac{1}{2}\sum_{i' \in l_j} J(x_i^T, x_{i'}^T)(\texttt{intuit}_{i,j} - \texttt{intuit}_{i',j})^2\right)$$

Making Proposals: $h_k(\texttt{score}^{\texttt{pred}}, \texttt{score}^{\texttt{intuit}})$ Our goal is produce a proposal of a feature-concept association, (i^*, j^*), that will both be highly predictive, and that the user is likely to accept. Meeting both of these conditions makes it likely that it will both be added to the model, and have a positive effect on predictive performance.

Recall that for a given proposal, the concept, j^*, is fixed. We compute the proposal by first taking the top k features in u_{j^*}–the features not yet probed by the algorithm for concept j^*, ranked by $\texttt{score}^{\texttt{pred}}$. The intuition is that any of these are predictive enough to help with downstream performance if the user were to accept the proposal. We then take

the highest ranking feature amongst these k ranked by $\texttt{score}^{\texttt{intuit}}$. This corresponds to choosing the term that the user is most likely to accept from amongst the predictive features found in the first step.

This thresholding and re-reranking allows us to incorporate both the association's predictiveness and the likelihood the user will accept it when making proposals.

4.5 Illustrative Toy Example

We introduce a simple, toy example to illustrate a scenario where neither $\texttt{score}^{\texttt{pred}}$ (pred) nor $\texttt{score}^{\texttt{intuit}}$ (intuit) alone make high quality proposals, but their combination, $h_k(\texttt{score}^{\texttt{pred}}, \texttt{score}^{\texttt{intuit}})$ (our method, pred-intuit), can.

Properties This example has 2 key properties that we expect to occur in real data. The first is that two distinct concepts have a similar effect on the prediction, which may be the case if, for example, both anxiety and insomnia have similar effects on psychiatric outcomes, even though they are 2 distinct clinical conditions. The second is that each concept consists of 2 groups of 2 highly correlated features, however these groups are not correlated. This may be the case when, a single condition may consist of 2 disjoint groups of patients, like patients with type I and type II diabetes, for example. Figure 4.2 shows the details of the toy example, where nodes that have the same color are closely correlated, edges for g have weight 1 and edges for f have their weight written on them. See Appendix Section B.1 for a detailed description of how the features were generated.

Results Figure 4.2 describes key failure cases for each variant. Black edges correspond to previously added associations, red edges correspond to the next 4 proposals made by the algorithm, and dotted red edges indicate there are multiple options for that proposal. The

numbers in the corresponding feature nodes indicate to the ordering of the red proposals. The intuit variant is able to successfully add the 2nd yellow node at the start of the 2nd concept when the concept was seeded with the first yellow node because these are clearly correlated. However for its second proposal, it can add any of the green, cyan or orange nodes nodes, since none of these are correlated with either the seed term for concept 2, or for any other concept. The pred strategy first proposes the green node, since this correctly labels the most additional instances with the concept, then proposes both of the cyan nodes because concept 3 (that the cyan nodes are associated with) plays the same predictive role as concept 2. After exhausting the cyan nodes, it returns to the second green node, however does not learn to do so immediately after the first rejected proposal, as our method does.

When run with 4 proposals for each concept, our method that combines both predictiveness and intuitiveness is able to recover the true set of features, and achieves almost perfect downstream and concept accuracy (concept accuracy: 0.994 ± 0.001, downstream accuracy: 0.990 ± 0.001), while the pred variant performs slightly worse on both metrics (concept accuracy: 0.978 ± 0.001, downstream accuracy: 0.966 ± 0.001), and the intuit variant performs even worse (concept accuracy: 0.892 ± 0.015, downstream accuracy: 0.858 ± 0.018). See Appendix Figure B.1.

4.6 Experiments

To allow for quantitative analysis and comparison to multiple baselines and variants of our approach, we ran experiments with known (hand-crafted) concepts to be discovered from real data: each experiment was seeded with features from the known concept, and we assumed that the simulated user would accept any proposed term that belonged to it.

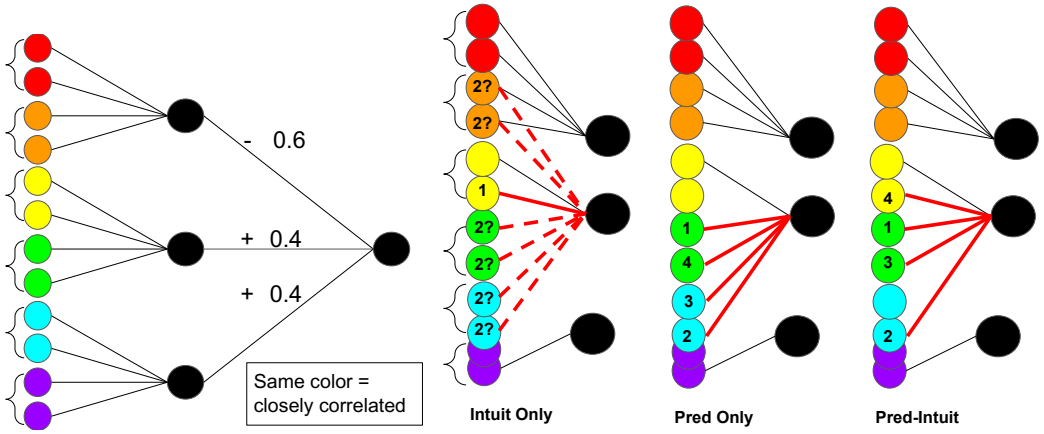


Figure 4.2: The toy example has 2 concepts that play similar roles in the prediction, and each concept depends on 2 distinct sets of correlated features. Black edges correspond to previously added associations, red edges correspond to the next proposals made by the algorithm, and dotted red edges indicate there are multiple best options for that proposal. The numbers in the corresponding feature nodes correspond to the ordering of the red proposals. The intuit variant can fill in the correlated sets of features, but is not effective at uncovering new sets of features, while the pred variant is unable to learn that features associated with a different concept that plays a similar predictive role are not relevant. Our approach (pred-intuit) avoids both of these pitfalls.

4.6.1 Setup

Datasets and Concept Definitions We use two domains: one publicly available dataset of Yelp restaurant reviews*, and one real, clinical dataset of patients diagnosed with depression from a Boston area hospital. In the Yelp data, we predict whether the average rating for a restaurant is good ($\geq$ 4 stars), or bad ($\leq$ 2 stars) based on counts of words in the aggregated reviews. In the Psych dataset, we predict whether a patient will be prescribed an atypical antipsychotic within 1 year of their first antidepressant prescription based on counts of the patient's past diagnoses, prescriptions and procedures. After preprocessing, the Yelp dataset has dimension 7,496x1,228, and the Psych dataset has dimensions 9,802x989; both are split 60/20/20 train/valid/test, and labels are class balanced by sub-

*https://www.yelp.com/dataset/

sampling. Neither of these real datasets come with concept definitions, so we crafted these concepts by hand to be realistic and reasonably predictive. The concepts in the Psych domain were informed by discussions with a practicing psychiatrist. See Appendix B.1 for additional details.

Baselines We compare to interactive, concept-based baselines as well as non-concept-based predictors. The concept-based baselines are comparable to our g (concept definitions), and have the same f as our method trained on top of them to make predictions. Since both approaches are interactive, they require an interaction method through which the user can provide feedback on the quality of the concepts–this is analogous to the step in our method where we propose a feature-concept association and the user provides feedback on whether it is intuitive. The first baseline uses active learning to fit a sparse, l1-regularized logistic regression classifier for each concept–denoted AL. Here, the feedback from the user consists of labeling instances with concept labels (whether the concept applies to that instance). The second baseline uses a supervised variant of anchor-topic-modeling[47], where the user feedback indicates whether a feature should be an anchor for a particular topic–denoted TM. Anchors are features that have a non-zero probability of appearing in only one topic, and can be used to guide the learned topics to take on a specific meaning. See Supplement Section B.2 for method details, hyperparameters, and results for an additional TM variant.

For non-interactive baselines that do not rely on concepts, we compare to a random forest classifier (RF), a neural network with a single hidden layer the of size C (NN), and l1-regularized logistic regression with a similar number of non-zero coefficients as C (LR). See Supplement Section B.2 for hyperparameters and details.

Finally, we explore variants of our approach: a pred variant that ranks proposals by $\texttt{score}^{\texttt{pred}}$, an intuit variant that ranks proposals by $\texttt{score}^{\texttt{intuit}}$, several variants of our method ('pred-intuit'), and several 'intuit-pred' variants that threshold based on $\texttt{score}^{\texttt{intuit}}$ then rank based on $\texttt{score}^{\texttt{pred}}$. Different variants correspond to different choices of k: 1%, 5%, 10%, and 25% of D. We denote these as e.g. pred-intuit-5 when $k = 5\%$ of D. k is not chosen by hyperparameter optimization since we do not have a validation set of concepts, instead we choose k=5% (pred-intuit-5) a priori to show in our main results as this value of k seems neither too restrictive, nor too broad.

4.6.2 Comparison to Baselines

The downstream accuracies and the concept accuracies on the test set are reported with standard errors in Table 4.1 for 25 randomly sampled sets of seed features from the list in Appendix B.1 and 10 proposals per concept.

	Yelp		Psych	
Variant	Downstream	Concept	Downstream	Concept
Ours	0.780±0.003	**0.707±0.002**	**0.606±0.002**	**0.778±0.010**
AL	0.753±0.003	0.612±0.003	0.581±0.003	0.598±0.013
TM	**0.811±0.009**	0.596±0.004	**0.611±0.003**	0.648±0.004
LR	0.696±0.000	-	0.588±0.000	-
NN	0.909±0.002	-	0.635±0.002	-
RF	0.935±0.001	-	0.670±0.001	-

Table 4.1: Downstream accuracy and concept accuracy ± standard errors for our method and baselines in both domains on the heldout test set. Bold numbers indicate the best accuracy(ies) for the concept-based methods. Our method performs substantially better on concept accuracy than concept learning baselines, while staying competitive on downstream accuracy. All interpretable baselines (above the horizontal line) have worse prediction than blackbox regressors (below line).

Our approach substantially outperforms all methods on concept accuracy. In Yelp, our final concept accuracy is 0.71 ± 0.0, and in Psych, it is 0.78 ± 0.1, 0.1 and 0.13 greater

TM			AL			Ours		
C1	C2	C3	C1	C2	C3			
managers	**neighborhood**	**creamy**	drivethru	**cozy**	**creamy**	**reservation**	**comfortable**	**juicy**
pull	**vibe**	**texture**	nicely	garbage	macdonalds		**welcoming**	**moist**
standing	playing	fruit	**reservation**	receipt	shift		**vibe**	**creamy**
bought	watch	vegas	lovely	macdonalds	addition		**casual**	
issues	bartender	**rich**	entrees	**comfortable**	perhaps		**ambiance**	

Table 4.2: Top 5 coefficients for each concept in AL, TM and Our method in Yelp (from restart with highest concept accuracy). Bolded features are related to the concept they are associated with. While for AL and TM some of the top features are relevant, many are irrelevant to the concept. For ours, the top (and only) features associated with each concept are all relevant by design.

than the 2nd best concept-based model respectively. These substantial differences suggest that our approach aligns much better with the user's intuitive representation than baselines, given a fixed number of user interactions. This is crucial for deriving valid interpretations of the predictive model in terms of these concepts.

Our approach is competitive with other concept-based approaches on downstream prediction accuracy. Our approach outperforms AL in both domains, and performs similarly to TM in the real clinical domain. However TM outperforms our approach in the Yelp domain, which may simply be a property of the Yelp data, as topic models are particularly well-suited to text data. These results suggest that our approach consistently performs well, including on real clinical data, that it is more effective than the AL baseline at optimizing downstream performance, and that it does not lose much on downstream accuracy compared to the TM baseline.

Our approach performs better on downstream accuracy than the LR baseline, which is comparable to training f directly on the raw features. In the Yelp domain, our approach outperforms LR by 0.08, and in the Psych domain, by 0.02. This suggests that training an interpretable model on top of the concepts constructed by our approach can boost predictive performance over training a sparse logistic regression on the raw features. Our concept based approach has other advantages as well, including that the predictors in

the interpretable model are user specified, whereas the inputs to a sparse logistic regression do not have any constraint on intuitiveness, and that colinearity[17] may cause weights on the raw features to be confused in e.g. sparse logistic regression, while the user chosen concepts are more likely to represent distinct ideas.

Standard predictors without interpretability constraints outperform all concept-based approaches on downstream accuracy. All the concept-based methods (including ours) have worse downstream accuracy than the non-interpretable methods (NN and RF); however, we emphasize that (a) neither of these baselines are interpretable and (b) there may be several ways to narrow that gap—the most substantial of which is moving beyond the particular concepts initialized by the user to find concepts that are both intuitive to the user, and highly predictive.

Comparisons against fully manual: Our approach outperforms the user manually selecting a small set of relevant features. We compare the downstream accuracy of our approach on the test set against randomly sampling features from the concept definitions to simulate a user generating g manually. The x-axis corresponds to the number of features sampled (without replacement) from each concept. Once all of the features for a given concept have been sampled, that concept is no longer updated. Figure 4.3 shows this for the 25 randomly selected sets of seed features used to generate Table 4.1. In Yelp, comparable downstream accuracy is never reached, and for Psych it takes approximately 50 features for some of the concepts to achieve comparable accuracy. The poor performance in the Yelp domain suggests that there are features that intuitively belong to the concept for the user, but that actively hurt predictive performance; our approach avoids adding these features. Overall, these results suggest that manually curating a predictive g will require much more

effort than seeding our approach with 1 relevant term.

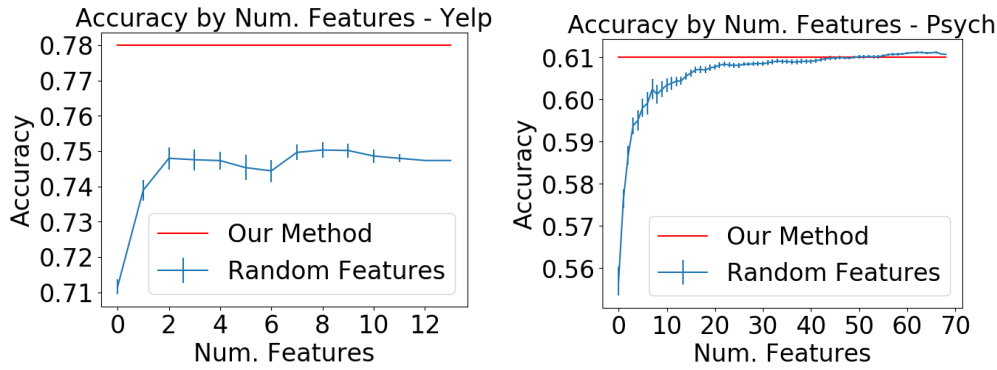


Figure 4.3: Heldout downstream accuracy by number of sampled relevant features. Yelp domain on left, Psych on right. In the Yelp domain, randomly adding features to the concepts from the list of "ground truth" related features never approaches the downstream performance of our method, while in the Psych domain, it takes up to around 50 random features per concept to surpass what our method achieves with 10 proposals per concept.

Spurious correlations in baseline concepts: Baselines have many irrelevant features in the top features for each concept; our approach, by design, does not. In Table 4.2, we report the top 5 features associated with each concept in the Yelp domain for both the TM and AL baselines. We see that, some of them are relevant to the topic at hand, for example 'vibe' for TM-C2, and 'comfortable' for AL-C2. Other examples are bolded in the table. However many of these features are not directly associated with the concepts, for example, 'vegas' in TM-C3, and 'addition' in AL-C3. These may lead to errors that could be caught by inspecting the model since it is interpretable, but unlike our method, AL and TM are not designed to allow the user to easily correct these errors.

4.6.3 Ablations and Variants

We repeat the same experiment as shown in Table 4.1, but exploring different variants of our approach, and show the downstream accuracy plotted against the concept accuracy on

the test set, as well as the number of accepted features for each variant in Figure 4.4.

The combination of predictiveness and intuitiveness to generate the proposal is important for downstream accuracy. Almost all variants that combine $\texttt{score}^{\texttt{intuit}}$ and $\texttt{score}^{\texttt{pred}}$ to make proposals outperform the pred and intuit variants on downstream accuracy. The left column of Figure 4.4 shows that all of the intuit-pred variants, and 3 of the 4 pred-intuit variants outperform pred and intuit (the 4th performs similarly). The last variant has a high value of k, thresholding after the top 25% of features. This is larger than it makes sense to set k to in practice, but demonstrates that even with large values of k, results are no worse than using only 'intuit' or 'pred'.

Both predictiveness and intuitiveness are also important for optimizing concept accuracy. Our proposed variant and several others outperform pred and intuit in concept accuracy, and no variants perform substantially worse. In Yelp, our proposed variant, pred-intuit-5% outperforms all other variants, and performs similarly or better to all variants in the Psych domain. This suggests that this variant is quite effective at optimizing both concept accuracy and downstream accuracy compared to other variants that use only $\texttt{score}^{\texttt{pred}}$ or $\texttt{score}^{\texttt{intuit}}$. We note that this particular setting of $k = 5\%$ of the total features was not chosen after hyperparameter optimization, but instead chosen a priori, however it seems to be a reasonable choice in both domains.

Pred-intuit variants, including our proposed variant make more accepted proposals than intuit-pred variants, potentially engaging users more effectively. The pred-intuit variants, including the one we propose, propose more accepted features than the intuit-pred variants. Both propose substantially more than the pred variant and less than the intuit variant. This is an important metric because even an approach that optimizes accuracy

well will likely not be engaging to the user if it rarely makes relevant suggestions. Given that intuit has much lower downstream and concept accuracy, our proposed variant of pred-intuit-5% balances these competing objectives well.

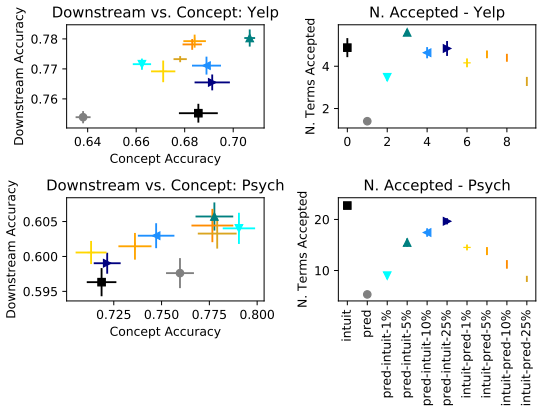


Figure 4.4: Results of our ablation study on the importance of both predictiveness and intuitiveness in the proposal. Variants that include both factors outperform pred and intuit in both concept accuracy and downstream accuracy, and our proposed pred-intuit variant tends to make more accepted proposals than intuit-pred variants. Our choice of $k = 5\%$ of D appears reasonable.

4.7 Evaluation of "Checks"

In evaluating this approach, we addressed some of the questions in our set of "checks" by either assumption or design, and others by running computational experiments with hand-crafted human proxies. In this work, we did not yet evaluate the system with human data to address the human-related questions with user data rather than computational proxies. In this section, we describe how we addressed the different checks, as well as future work related to user evaluation.

We address HQ1 by assumption, assuming that giving feedback on these feature-concept associations is not too cognitively challenging. This assumption aligns with the findings

in Raghavan et al.[66] that found that labeling a feature takes a fraction of the time of labeling an instance. HQ2, we address partly by an assumption that merits further testing, and partly by evaluation with proxy feedback. If we assume that concepts defined using the functional form g and people's internal representation of concepts are closely related, then the task, by design, provides direct feedback on this, addressing HQ2. This is in contrast to the anchor topic model baseline where the feedback less transparently impacts the learned concepts. However, this assumption that g is expressive enough to capture human concepts merits further testing.

HQ3 and HQ4 we address through experiments with computational proxies. We can think of the correlation-only baseline as a human proxy where the user chooses only closely correlated features to add to the concept. Based on this proxy, we find that our approach performs better, however further work should test these questions with real human data based on this promising outcome.

MQ1, MQ2 and MQ3 we address by evaluation with hand-defined proxies of specific concepts defined according to the functional form of g. These were all evaluated through the direct evaluation of the approach compared to baselines (topic models and active learning as well as ablations). Based on the good performance of our method compared to these, we can assume that it performs relatively well with respect to these 3 questions.

Finally, MQ4 we address by assumption, as the concepts are, by definition, required to align with human representations. These are not computationally encoded, and so require human feedback. One can also consider the signal that not all proposals are accepted as evidence that the user oversight is required to complete this task well.

4.8 Discussion

Our approach offers advantages including providing the means to create a prediction system that allows people the ability to reason about the prediction process at a higher level of abstraction than the raw data, transparent concept definitions that can be validated for intuitiveness, and a feedback mechanism that allows for efficient use of human feedback. However it also has limitations that suggest interesting future work.

On our two tasks, all interpretable, concept-based approaches came at a cost to downstream accuracy. Future work could address this by making g more flexible while still ensuring that its intuitiveness can be validated. Alternatively, this could be addressed by seeding our approach with concepts that are both intuitive and highly predictive in order to mitigate potential performance effects. The second option may also address cases where the user cannot easily specify a set of predictive concepts and an associated feature.

While our results show good performance in realistic simulations of human feedback, we did not test it extensively with users. There remain important open questions including: how well can g capture the high-level concepts in a variety of domains? And how accurately are people able to give the accept/reject feedback required by our algorithm? For the second, it may be possible that the user believes a feature-concept association is intuitive, but it actually makes the concept definition less representative of the user's concept. How realistic this scenario is is an interesting question for future work.

4.9 Conclusion

We proposed an approach for learning interpretable concept-based latent representations to extend interpretable machine learning methods to domains with uninterpretable features. We used human-in-the-loop training to learn transparent concept definitions that align with users' intuitive representation of a prediction problem, and we showed on real datasets with simulated concepts that our approach can learn representations that align substantially better with user-intuitive concepts.

Transparent machine learning methods allow users to inspect system logic, potentially catching mistakes and improving models. Our approach scales these benefits to high-dimensional domains with unintuitive features without sacrificing transparency at the representation level.

5

Method 3: Optimizing Elicitation of Human Context at Test-Time

In this chapter, we address the problem of missing human context that may be obvious to the user for a particular test instance, however that the ML model does not have access to. We formalize this problem as an unobserved "human feature set" that must be queried at a

cost for test instances. Under our framework, h is the values of these missing features, and we choose q based on a function that combines uncertainty about h for different features, and the importance of h for the prediction outcome. The goal is to make accurate predictions.

5.1 Introduction

ML models and human decision-makers often have different yet complementary strengths that can be leveraged to improve human-ML decisions. The specific case we focus on here is that ML models can recognise patterns in massive datasets, while humans may have access to additional facts and contextual information which may be missing from the dataset and are important for decision-making. Methods exist for incorporating human expertise during model training–e.g. Bayesian methods and the methods described in Chapters 3 and 4, however incorporating expertise of this type at test time allows additional flexibility for personalizing queries to different users.

However, existing strategies for incorporating human expertise into ML predictions at test-time largely rely on users being able to make expert decisions. For example, mixtures-of-experts approaches (e.g. [30], [56]) weight the predictions made by an ML model and a human decision-maker to decide when a human should step in. Other approaches learn when to defer decisions to a down-stream decision-maker based on samples of the ML model's decisions (e.g [50], [53], [31]), or explicitly constrain the model to align with human expertise [64], [91]. Finally, explanations (e.g. [92], [40]) allow users to synthesize the ML model's insights with their own, but require the difficult cognitive task of deciding how to reconcile these insights. These methods all assume that users have substantial expertise about how to make deci-

sions in the domain, however other important types of human expertise exist. We consider the case of non-expert users who may have access to relevant contextual information about a particular instance.

To illustrate this setting, consider the example where electronic health record (EHR) data from thousands of patients is converted to a set of machine features that are analysed by ML and used to predict depression-related outcomes (e.g.[63,38]). These models, though predictive, may lack important contextual information that is otherwise available to the human patient, about, for instance, their support network, or socioeconomic status. We call these human features, and *both human and machine* features can be important predictors of treatment outcomes[5]. Importantly, these human features may consist of information available to the person *about whom* a prediction is made (e.g. the patient) rather than the person with prediction expertise (e.g. the doctor). This human context information is the h defined in our framework.

However incorporating these human feature-level insights into ML predictions requires formalizing this problem in a way that permits ML optimization. The first contribution we make in this work is to contrast the problem of feature-level synthesis of human and ML decision making to existing approaches. The second is to formally describe this problem as intelligently querying a unique set of human features from a known feature set for each instance at test time, given a model trained with all features (we assume weights for the human features were learned during training)–operationalizing q from our framework [*].

[*]Having access to these features at train time may be reasonable in settings where there is a budget for collecting additional human features at train time, including through crowdsourcing (e.g.[13]), or in clinical trials (e.g. the CoMMpass study for multiple myeloma https://themmrf.org/finding-a-cure/our-work/the-mmrf-commpass-study/). But expecting these same features at test time might be infeasible for either financial or workflow-related reasons. For example, asking a patient 100 questions during a clinical interview may negatively impact the clinical relationship.

This differs from feature selection as we are interested in eliciting a personalized subset of human features for each instance at test time, rather than a global set of human features for all instances. This is important, as different features may be relevant for different types of patients. For example, marital status may be important to determine social support for a 40 year old patient, but not an 18 year old.

The third contribution we make in this work is to demonstrate on real datasets with realistic human-machine feature splits that this problem formulation leads to substantially improved predictive performance over only using machine features, or querying only a global set of features using standard feature selection methods. We do this based on an active-learning method introduced in [65] that we demonstrate effectively solves the optimization problem we propose for the feature-level synthesis of human+ML decisions problem. Importantly, we also demonstrate an improvement of the method in [65] over baselines in real datasets, something the original paper does not do. We also describe a heuristic addition to this approach that proves important for consistently outperforming baselines.

Finally, we seek to understand how the query method we apply works by proposing 3 mechanisms for its behavior, and 3 corresponding heuristics to solve the problem of querying features under each of these mechanisms. We demonstrate through simulations that these heuristics solve the problem when that mechanism is at play, however we also demonstrate in our real-data examples that none of these heuristics match the performance of the method we study. This suggests that it is doing something more complex than simply addressing one of these mechanisms, which also suggests that it may be difficult for a user to solve this problem rather than relying on this computational approach.

5.2 Illustrative Example of Feature-Level Human+Machine Synthesis

The problem of combining human and machine insights to make better predictions has been well studied, however existing approaches do not consider the case of feature-level synthesis that we propose. Importantly, this approach allows different users with different types of insights to contribute to predictions, as the user must have access to some additional features, but is not required to be an expert decision-maker. We demonstrate through a toy example how feature-level synthesis can produce correct predictions in cases where existing approaches with a non-expert user would fail.

Specifically, we compare feature-level synthesis to methods requiring predictions from the model and human to combine–we call these "post-hoc prediction combinations," and methods that provide users with an explanation of the model's prediction and expect them to incorporate this information cohesively into their prediction. Both of these are challenging for users without substantial domain expertise, and produce incorrect predictions in this simulated scenario, while feature-level synthesis corrects those errors.

5.2.1 Post-Hoc Prediction Combinations

Various methods exist for combining the predictions of multiple predictors to improve performance. Relevant in this setting are mixtures of experts when one expert is a human (e.g.[64,56]), and methods that triage predictions between human and machine decision-makers (e.g.[50,67,93]). While these are quite effective in some settings, they can fail in cases where no individual decision-maker makes a correct prediction. This may often be the case when the human user is not a domain-expert, as in the case of a patient with relevant knowledge about their own context, but no formal medical training. Even with domain-

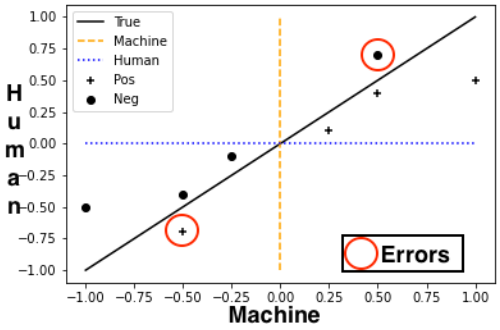


Figure 5.1: This figure shows a toy dataset with a human feature (y axis) and a machine feature (x-axis) that are both needed to produce the true decision boundary (solid line). The orange dashed line ($x = 0$) and the blue dotted line ($y = 0$) are the best fit lines based on the human and machine feature respectively. Both mis-classify the 2 red-circled points, but a model that is able to use both features can classify perfectly along $y = x$.

expert users, there may be cases where neither they nor the ML model make a correct prediction. Synthesizing insights at the feature level can allow for correct predictions even in these cases.

We illustrate this through a pedagogical toy example of a binary classifier with 2 features– one of which belongs to the machine feature set, and the other to the human feature set. Figure C.2 shows this example, where the true labels depend on both features, and the best-fit lines for either feature alone mis-classify the same 2 points. Any weighted average[†] of these 2 wrong predictions would still be wrong, which means that this situation cannot be addressed with a post-hoc combination of the predictions. On the other hand, if both features were available to the model when making the prediction, a linear classifier can easily separate the points perfectly. This demonstrates that feature-level synthesis can have important advantages compared to decision-level synthesis.

[†]We consider the case of weighted averages as this is assumption in mixtures of experts, and the triaging setting is a specialized case of this where one or the other prediction is chosen.

5.2.2 Explanations

Interpretable ML models that provide explanations (e.g. [92], [40]) allow for synthesis of ML and human insights by allowing the user to understand the ML model's prediction process and synthesize it with any of their own insights. However there are scenarios where this may be challenging for the user, particularly if they are not an expert at the prediction task.

In the pedagogical example in Figure C.2, for example, the ML prediction process is interpretable (i.e. predict 1 `if` $x > 0$), however it is not trivial to build on this insight to improve the predictions. A straightforward combination of that explanation and the human's decision boundary by taking the "or" of these decision rules is: predict 1 `if` $(x > 0$ `or` $y > 0)$. This does not address the errors circled in red. The same is true if we combine the human and machine decision rules with "and."

Instead, fixing the erroneous predictions requires generating a completely new decision boundary using both the human and machine features, which is a more cognitively challenging process that requires substantial understanding of the decision problem. This is not a process that would be accessible to someone who is not an expert decision-maker.

5.3 Problem Definition and Approach

The key novelty in our problem definition lies in presuming the existence of a set of "human-features" that can be elicited at test time, and that can improve prediction quality. Our core intuition is that this problem can be formalized as the problem of how to query a small, but useful set of human features for each instance, given that we have some way of making predictions based on them. We first formally describe this problem, then we demonstrate

that the problem is solvable by describing how the approach introduced in[65] provides an effective solution.

5.3.1 Preliminaries

Concretely, we assume a standard supervised learning setup where we have a dataset of N instances, the machine features, X^m, that the machine learning model always has access to, and corresponding labels, Y.

In addition, we assume access to human features X^h during training but not at test time when they must be queried for a cost. We assume the user has access to these features for a specific instance, which may be the case when the instance corresponds to a representation of the user–for example their medical history. We assume that, for each instance, we can query the associated user a small, fixed number of times B, for e.g. less than 10 times, for the values of specific features. More queries may overwhelm the user. Finally, we assume that a distinct set of human features may be relevant for each instance. For example, demographic characteristics such as marital status exist for all patients, but may be less relevant for indicating social support in an 18 year old patient than a 40 year old.

X^m and X^h could be any real values, but in our instantiation, we assume X^h is binary: $X^h \in \{0,1\}^{N \text{x} D_h}$. (The human is best at providing yes-no answers.) We additionally assume that the labels are categorical, i.e. $Y_n \in \{0, \ldots, K\}$.

We train a discriminative prediction function f to make predictions $\hat{Y}$ as follows:

$$\hat{Y} = f(\theta^m, X^m, \theta^h, X^h) + \varphi \tag{5.1}$$

where θ^m and θ^h are parameters for X^m and X^h respectively and φ is a bias term. We instanti-

ate the model class when discussing implementation, but the methods and problem definition apply generally to models of this form. Fitting these parameters is not the focus of this work.

5.3.2 Defining the Desired Optimization Objective

We formalize the problem of feature-level synthesis of human+ML predictions as a constrained optimization problem where the goal is to intelligently select B human features to query at test-time specific to each instance that maximize the predictive quality, L, for that instance.

For a test instance x, we denote which features in x^h have been queried, using a binary query mask, $q \in \{0,1\}^{D_h}$. $q_d = 1$ indicates that human feature d has been queried for this instance, while $q_d = 0$ indicates that it has not been queried.

We can then write the constraint on the number of queries for each instance in terms of q:

$$\sum_{d=1}^{D_h} q_d \leq B \tag{5.2}$$

In other words, the number of elements in q must that are queried, i.e. set to 1, should be at most B.

To make predictions on a test instance where we only have access to queried features in q, we marginalize out the remaining features (i.e. $q_d = 0$). This is a standard way of handling missing data. We denote this prediction function as f^{marg}, and use it to predict for an instance as follows:

$$f^{marg}(x^m, x^h, q) = \int_{\tilde{x}^h} f(\theta^m, x^m, \theta^h, \tilde{x}^h) p(\tilde{x}^h | x^m, q) \tag{5.3}$$

where $\tilde{x}^b$ corresponds to a random variable sampled from the helper distribution $p(\tilde{x}^b|x^m, q)$ defined below, rather than the actual value of x^b, as this is unknown.

To define this helper distribution $p(\tilde{x}^b|x^m, q)$, first factorize it, as x^b and x^m are both high dimensional, and thus challenging to model jointly. The factorized form is as follows:

$$p(\tilde{x}^b|x^m, q) = \prod_{d=1}^{D^b} p(\tilde{x}^b_d|x^m, q_d) \tag{5.4}$$

We then define the individual dimensions of the helper distribution as follows:

$$p(\tilde{x}^b_d|x^m, q_d) = \begin{cases} 1_{\tilde{x}^b_d = x^b_d} \texttt{ if } q_d = 1 \\ p(\tilde{x}^b_d|x^m) \texttt{ if } q_d = 0 \end{cases} \tag{5.5}$$

I.e. the probability is the same as $p(\tilde{x}^b_d|x^m)$ if the feature hasn't been queried, and if the feature has been queried, the probability of the true value is 1, and zero otherwise.

In practice, we compute the expectation in Equation 5.3 by Monte Carlo sampling:

$$f^{marg}(x^m, x^b, q) \approx \frac{1}{S} \sum_{s=1}^{S} f(\theta^m, x^m, \theta^b, \tilde{x}^{b(s)})$$
$$\tilde{x}^{b(s)} \sim p(\tilde{x}^{b(s)}|x^m, q)$$

where S is the number of samples drawn.

Based on this, we can define an idealized optimization objective that optimizes q for instance x to minimize predictive loss L between the true label y and the prediction made using only the machine features and the queried human features in q. This objective can be

written as follow:

$$q* = \underset{q \in \{0,1\}^{D_b} \text{ s.t. } \sum_{d=1}^{D_b} q_d \leq B}{\operatorname{argmin}} L(y, f^{marg}(x^m, x^b, q)) \tag{5.6}$$

In our results, we operationalize L as f1-score, but other reasonable choices could be made.

5.3.3 Approximate Solution to Desired Objective

Solving this optimization problem directly requires access to the true label, y, which we do not have for test instances. A such, we use an active learning method proposed by[65] to solve this problem. This approach iteratively selects the query d^* for an instance that minimizes the expected entropy of the marginalized predictions. We give a brief description of this approach for completeness, but for a more thorough description, see[65].

For each query for each instance, this approach solves the following optimization problem:

$$d^* = \underset{d \in \{1,\ldots,D^b | q_d = 0\}}{\operatorname{argmin}} \mathbb{E}_{p(x_d^b | x^m)}[H(f^{marg}(x^m, x^b, q + 1_d^{D^b})] \tag{5.7}$$

where $1_d^{D^b}$ denotes a onehot vector of size D^b where all entries are 0 except at index d where the entry is 1. The expectation is easy to compute as we assume x_d^b is binary.

Intuitively, minimizing the entropy finds the human feature to query that will reduce our uncertainty about the prediction as much as possible. In the absence of true labels, prediction uncertainty gives a reasonable surrogate for Equation 5.6. Taking the expectation of the entropy w.r.t the missing human features is necessary, as these feature values are unknown, by assumption, when considering whether to query them.

We solve the full optimization problem by greedily repeating this procedure for each

query in the budget B. While this is not guaranteed to provide an optimal solution, it is often employed for similar approaches, e.g.[25], and is generally quite effective. In practice, we solve the optimization subproblem described in Equation 5.6 by trying all possible values of d that satisfy the constraint. This is a strategy linear in D^b, and is employed by many active learning approaches, including[99]. Algorithm 2 gives a full description of our procedure.

Algorithm 2 This greedy search-based optimization procedure chooses, at each iteration, to query the human feature that minimizes the expected marginalized entropy given the feature is queried.

Given: x^m, x^b
$q \leftarrow \{0\}^{D^b}$
for $\mathsf{b} \in \{1, \ldots, B\}$ **do**
  $d^* = \text{argmin}_{d \in \{1,\ldots,D^b | q_d = 0\}} \mathbb{E}_{p(x_d^b | x^m)}[H(f^{marg}(x^m, x^b, q + 1_d^{D^b})]$
  $q \leftarrow q + 1_{d^*}^{D^b}$
end for

5.4 Implementation

In order to solve the problem we introduced in previous section, we need to instantiate a functional form for the prediction function f, and a distribution for approximating $p(x^b|x^m)$. We also describe a heuristic approach to boosting predictive performance by re-training the parameters of the predictive function given the learned query mask, q. These instantiations and heuristics are novel additions to the approach described in[65] and lead to substantially improved results in our experiments where we demonstrate that this approach outperforms baselines on real datasets–an insight that was missing from[65].

5.4.1 Defining a functional form for f

In practice, we implement f as a logistic regression model, where we make predictions as follows:

$$\hat{Y} = \sigma((\theta^m)^T X^m + (\theta^b)^T X^b + \varphi) \tag{5.8}$$

Where θ^m and θ^b are weight vectors in $\mathbb{R}^{D_m}$ and $\mathbb{R}^{D_b}$ respectively, and φ is an intercept term in $\mathbb{R}$.

5.4.2 Approximation $p(x^b|x^m)$

The marginalized prediction function, f^{marg}, and the expectation in Equation 5.7, depend on $p(x^b|x^m)$. We approximate this quantity from the data at train time. In order to fit the approximation $\hat{p}(x^b|x^m)$, we model each dimension of x^b with an independent logistic regression model, as the features in x^b are binary. For a dimension d, we define:

$$\hat{p}(x_d^b|x_d^m) = \sigma(w_d^T x_m + w_d^0) \tag{5.9}$$

where w_d is a weight vectors in $\mathbb{R}^{D_m}$, and w_d^0 is an intercept in $\mathbb{R}$. We train parameters w_d and w_d^0 to minimize the log-loss between $\hat{p}(X_d^b|X^m)$ and X_d^b for each dimension $d \in \{1, ..., D^b\}$ using the training set. While more expressive approximations to this distribution are possible, we found this one to be sufficient in practice.

5.4.3 Implementation heuristic: Re-training with Query Mask

We describe a retraining heuristic that boosts performance. The parameters of the prediction function, θ^m, θ^b, φ are trained assuming that the dataset includes the human features

in addition to the machine features. We address this by marginalizing out unqueried dimensions when predicting, but we can do better in practice by re-training the parameters using query masks optimized for the training set.

In this case, rather than marginalizing the unqueried features to make predictions, we zero them out. We define a prediction function f^{zero} as follows:

$$f^{zero}(x^m, x^b, q) = ((\bar{\theta}^m)^T x^m + (\bar{\theta}^b)^T (x^b \odot q) + \bar{\varphi}) \tag{5.10}$$

and fit the re-trained parameters $\bar{\theta}^m$, $\bar{\theta}^b$, and $\bar{\varphi}$ using a standard approach on the transformed dataset: $(X^m, (X^b \odot Q), Y)$. Q is the matrix of feature query masks fit using the approach described in Section 5.3.3 on the training set. At test time, we make predictions using f^{zero}.

5.5 Experiments

We ran experiments simulating our proposed problem setting in 2 real domains, where we focus on 3 key research questions: 1) Can a small set of human features substantially improve performance over a machine-only baseline? 2) Does querying features *for each specific instance* lead to performance improvements over global feature selection from the human features? 3) Qualitatively, why does querying different features for each instance matter? We use the approach described in [65] and operationalized for our problem setting to answer these questions.

Our results demonstrate that the instance-specific queries to a human feature set can substantially improve performance, and that the method in [65] with our retraining heuristic

provides an effective solution to this problem. Finally, in Section 5.6, we provide additional results comparing this approach to some simple heurstics to better understand where the performance gain comes from. We also include qualitative results and results on an illustrative toy example in Appendix B.

5.5.1 Experimental Setup

Baselines We compare the proposed solutions to two baselines and one oracle upper bound. The upper bound consists of a model with the same functional form as f (i.e. logistic regression) trained using both the machine and human feature sets, i.e. X^m and X^h–we denote this *all-features*. This would be what would be possible if, at test time, we could query the human for *all* their features rather than a subset. The first baseline consists of another logistic regression model trained with only the machine feature set, i.e. X^m, we denote this *machine-only*. This is the performance of the model with no access to human features. The second baseline consists of a standard feature selection (denoted *feature-selection*) approach where the same B human features are queried for all instances, rather than a distinct subset of human features being queried for each instance. This allows us to demonstrate the importance of instance-specific human feature queries.

We implement the feature selection baseline using a greedy forward selection strategy (see[84] for an overview) where, for each of the B features to add, we re-train the model adding each remaining human feature and choose the feature with the best validation performance (computed using f1-score, as this is the metric used in our results). This is a standard and effective approach to feature selection.

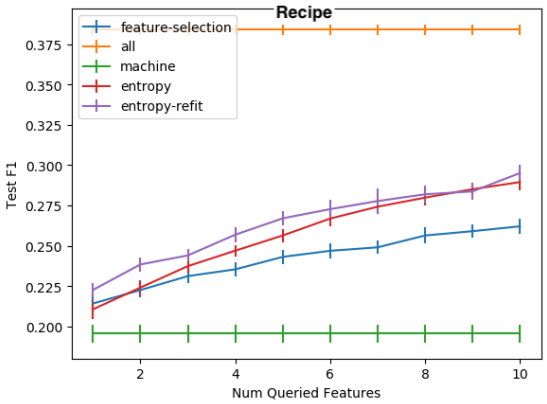


Figure 5.2: Test f1-score as a function of B for both instance-wise feature selection methods, baselines and upper bound in *recipe* dataset. Error bars are standard errors over 10 random restarts. *all-features* performs best and *machine-only* worst with the methods using a subset of human features in between. *entropy-selection* and *entropy-retrain* both substantially outperform *feature-selection*, demonstrating the importance of instance-wise feature selection.

HYPERPARAMETERS We set $B = 10$ as this is a manageable number of features for a single user to provide, and the number of Monte Carlo samples $S = 5000$ to minimize this source of approximation error. We select hyperparameters to maximize F1-score (the metric we report) on the validation set, except in the case of the models for $p(x^b|x^m)$, where log-loss is minimized instead. This is to encourage those models to accurately model the probability distribution they approximate. See Appendix I for additional hyperparameter details.

DATASETS To illustrate the performance of our proposed approach, we use two real datasets: a *recipe* dataset[‡] where the goal is to predict the cuisine of a recipe (e.g. Italian food) based on the ingredients, and a *birds* dataset [Wah] where the goal is to predict the type of bird (e.g. crow) based on some crowdsourced attributes of an image of a bird. After pre-processing, the *recipe* dataset consists of 6k instances (subsampled to that size for computa-

[‡] https://www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset train.json file

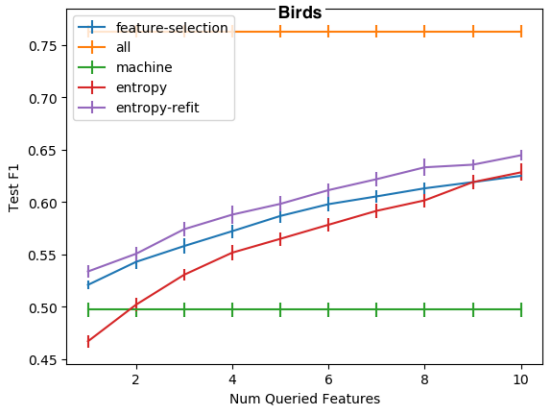


Figure 5.3: Test f1-score as a function of B for both instance-wise feature selection methods, baselines and upper bound in *birds* dataset. Error bars are standard errors over 10 random restarts. In both, *all-features* performs best and *machine-only* worst with the methods using a subset of human features in between. *entropy-retrain* eventually outperforms baselines, but *entropy-selection* performs slightly worse, demonstrating the importance of our retraining heuristic.

tional reasons); 120 features, and 20 classes, and the *birds* dataset consists of 5k instances; 171 features; and 36 classes. See Appendix 1 for additional pre-processing details.

We split the feature sets between X^m and X^b to facilitate running experiments with ground truth values of X^b. We construct these feature splits so that X^m consists of a smaller set of features that are "simpler" in some way, and X^b consists of a larger set of features that are more "complex". In the *recipe* dataset, we split the features so all single word ingredients (33 features) are in the machine feature set, and all 2+ word ingredients (87 features) are in the human feature set. For the *birds* dataset, we split them so all non-color-related words (48 features) are in the machine features, and all color-related words (123 features) are in the human features.

5.5.2 Results

In our experiments, we compared the predictive performance of the vanilla method proposed in[65] (*entropy-selection*) and our retraining heuristic described in Section 5.4.3 (*entropy-retrain*) against baselines and the *all-features* upper bound. We report test f1-score for all of these across values of B. Results are reported as the mean and standard deviation over 10 random restarts of the train/valid/test splits and any subsampling. Figures 5.2 and 5.3 show results in the *recipe* and *birds* domains respectively.

The full set of features outperforms *machine-only*, suggesting that the human features contribute substantially to predictive performance. In both datasets, *all-features* performs substantially better than *machine-only*. In the recipe dataset, the gap in f1-scores between these methods is 0.18, while in the birds dataset, the gap is 0.26. This suggests that there is substantial improvement to be gained from finding ways to use the human features in predictions.

All 3 approaches that query human features are able to close a substantial portion of the gap between *all-features* and *machine-only* within just 10 queries. In both datasets, the *entropy-selection*, *entropy-retrain* and *feature-selection* methods all substantially improve performance over *machine-only*. The 2 entropy-based methods we propose close half or more of the performance gap by the 10th query in both datasets, while *feature-selection* only achieves this in the *birds* dataset. This is true despite the 10 queries covering only 12% of the human features in the *recipe* dataset and 8% of the features in the *birds* dataset. This suggests that querying a small number of relevant hu-

man features is a viable strategy to substantially improve performance of ML models that have access to only a subset of relevant features.

entropy-selection has variable performance compared to *feature-selection*, outperforming it in some cases and underperforming it in others. In the recipe dataset, *entropy-selection*, performs similarly to *entropy-retrain*, which allows it to substantially improve over *feature-selection* starting on the 4th query. In the birds dataset, *entropy-selection* actually performs worse than *feature-selection* for the first 6 queries, then performs similarly. This suggests that the vanilla method proposed in[?] without the retraining heuristic is promising, but not always effective–at least given a limited number of queries. See Appendix C.1 for a follow-up experiment showing that the entropy method eventually outperforms the baseline after an initial round of *feature-selection*.

Our proposed *entropy-retrain* approach outperforms *feature-selection* in both datasets after sufficient queries, and never underperforms it. In the *recipe* dataset, *entropy-retrain* outperforms *feature-selection* starting from the 2nd query onwards. In the *birds* dataset, *entropy-retrain* performs comparably to *feature-selection* for the first 6 queries, then outperforms it for the last 4. The performance improvement is particularly marked in the recipe domain, where *entropy-retrain*'s f1-score after 10 queries is 0.3 compared to only 0.26 for *feature-selection*. This suggests that querying unique human features for each instance can improve performance over a shared set of features queried for the entire dataset. It also suggests that our proposed retraining heuristic in *entropy-retrain* is important for achieving these performance gains.

5.6 Understanding Why this Works

In the experiments above, we demonstrate that the problem formulation we put forth is potentially impactful, as querying these instance-specific human features improves performance, particularly over a feature-selection baseline that chooses the same features for all instances. However, these results do not help us understand why the method introduced in[65] that we apply to solve this problem works. More importantly, can we make any guesses about how this method would compare to an approach where we ask users directly for the features they feel are relevant rather than computationally suggesting them? While fully answering this questions requires a user study, which we do not do in this work, we provide some preliminary results towards answering this question by proposing some plausible mechanisms for the method we use; deriving some heuristics based on those methods that a user might be able to apply in practice; and demonstrating that in our real-data examples, these heuristics are not sufficient to solve the problem as effectively as the entropy-based acquisition function.

5.6.1 Proposing Mechanisms for Entropy-Based Feature Acquisition

We propose three mechanisms through which the entropy selection approach may be working. For each of these, we describe a data generating process where this mechanism would be a sensible approach to the problem.

Mechanism 1: Globally important features The first mechanism we consider is that entropy selection works by choosing features that are globally important across the dataset. This corresponds to choosing features that are highly predictive for all instances

to query, rather than doing something more instance-specific. A data generating process where this would be a reasonable approach is the case where X_m and X_b are generated independently, however both contribute to predictions. In this case, since X_m doesn't provide us with any information about X_b, the best we would expect to be able to do for queries is to choose features that are important on average.

Mechanism 2: Refining plausible classes The second mechanism we consider is that X_m provides enough information to narrow down the set of plausible labels (as we consider multi-class prediction problems), and then the entropy approach chooses predictive features that differentiate between that set of plausible labels. Additional qualitative results in Appendix C provide some evidence suggesting this may be happening in the recipe domain. A data generating process where this would be a reasonable approach is the case where X_m is drawn from the same distribution for small groups of class labels (for example, French and Italian food), then X_b is drawn with a subset of dimensions having high probability that differentiate between the group of plausible class labels. In this case, X_m focuses us in on a set of plausible classes, then X_b allows us to distinguish accurately between those options.

Mechanism 3: Proposing surprising features The third mechanism we consider is that X_m provides information to accurately guess much of X_b, but that depending on the values of X_m, some dimensions of X_b may be a surprise. For example, a recipe that uses garlic is also likely to use onions, and a recipe that uses cream cheese frosting is unlikely to use onions, making the question of whether a recipe uses onions unlikely to be a surprise if we know it uses garlic or cream cheese frosting. On the other hand, if a recipe uses butter,

it may or may not use onions, making the feature value of onions surprising in this context. This mechanism suggests that we may want to know the values of features that we cannot guess based on X_m. A data generating process where this would be a reasonable approach is the case where different values of X_m reduce uncertainty about different dimensions of X_b, and then X_b is drawn according to these probabilities.

5.6.2 Proposing Heuristics to Address Mechanisms

We lay out a heuristic to address each of these proposed mechanisms. We first argue why these are plausible heuristics for how someone might approach solving this particular problem, then demonstrate on toy-data simulations that these heuristics work when there is a match between the mechanisms needed by the dataset, and the heuristic.

Heuristic 1: Feature Selection The heuristic we propose to address the first mechanism is the feature selection baseline we study in the previous set of results. This baseline makes a global choice of which dimensions of X_b to query based on which are generally most predictive overall. This is a good strategy in the case of mechanism 1, where X_m does not provide sufficient information to personalize beyond this.

Heuristic 2: Feature importance for top classes The heuristic we propose to address the second mechanism is to first define a set of plausible predictions based on $p(y|X_m)$, then to select the most important features related to only these classes. This is a reasonable heuristic in the case of the data generating process described with mechanism 2 where X_m allows us to narrow in the prediction to a set of plausible options, and then the role of X_b is to choose between these plausible options. We implement this by identifying

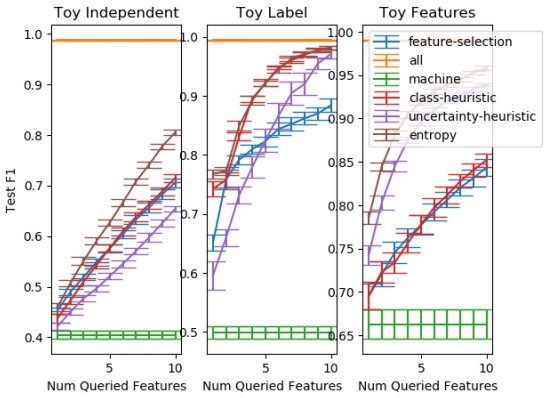


Figure 5.4: Test f1-score as a function of B for the vanilla entropy approach and the 3 heuristics. Each plot corresponds to a dataset designed based on a different of the 3 mechanisms (1 to 3 left to right). In each case, the corresponding heuristic performs as well or better than the other 2 heuristics, as expected. The entropy approach performs at least as well as the best heuristic in each case as well.

plausible classes as those where $p(y_i|X_m) > 1/K$ where K is the number of classes, and so $1/K$ is the uniform probability. We then rank features by the largest magnitude of any weight related to this feature in the set of weigths for these plausible classes.

HEURISTIC 3: FEATURE UNCERTAINTY The heuristic we propose to address the third mechanism is to choose the features to query with the highest uncertainty based on the $p(X_b|X_m)$. This addresses the case in mechanism 3 where X_m allows us to predict some aspects of X_b very well (and perhaps different dimensions of X_b depending on X_m), but that other dimensions remain uncertain. In this case, knowing the feature values of these uncertain dimensions will help us refine our estimates of X_b used in prediction. We operationalize this as sorting features to query by the magnitude of the deviation of their probability given X_m from 0.5. I.e. $|0.5 - p(X_b^i|X_m)|$.

Generating toy datasets "Toy independent" is generated according to mechanism 1, and has both X_m and X_b generated independently from Bernoulli(0.1). y is generated from a

random logistic regression model that we describe in more detail below. "Toy label" is generated according to mechanism 2 and has X_m generated independently from Bernoulli(0.1) as before. We then generated grouped labels in group labels g using X_m that indicate that an instance belongs to one of K specific classes. I.e. we generated groupings that tell us, for example, instance 1 is label A or B, and instance 2 is label E or F. Then, we generate X_h based on g such that each dimension of X_h is matched to a dimension of g, and if the label belongs to that grouping for an instance the probability that the X_h dimension will have value 1 is 0.5, and otherwise it is 0.01. We then generated y from a random logistic regression model based on X_m and X_h. Finally, to generate "Toy features" for the features mechanism, we generated X_m as in the other 2 cases. Then we generated X_h based on X_m such that each dimension of X_h is matched to a dimension of X_m, and if that X_m dimension has value 1 for an instance the probability that the X_h dimension will also have value 1 is 0.5, and otherwise it is 0.01. In practice, this means that observing the relevant X_m makes the X_h value highly uncertain, whereas otherwise it is most likely negative. We then generated y from a random logistic regression model based on X_m and X_h. To generate the random logistic regression outputs, we generate random weights from a normal distribution with mean 1. and variance 0.1. We repeat this process until we find weights that generate an approximately even distribution over outcomes–i.e. the minimum class prior probability is within 0.05 of the uniform distribution. For each dataset, we generate 10000 samples with 10 machine features, 20 human features, 6 classes, and 3 groups of size 2.

Heuristic validation Figure 5.4 shows the results of the 3 heuristic-based approaches and the entropy approach in each of the toy datasets. In each case, the corresponding heuristic performs as well or better than the other 2 heuristics. In the label and feature cases, the

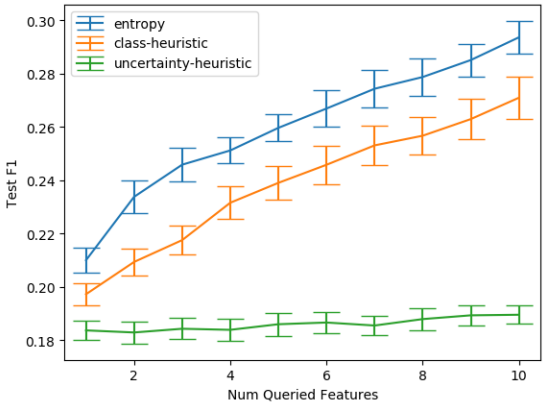


Figure 5.5: Test f1-score as a function of B for both the entropy approach, and proposed heuristics 2 and 3 (heuristic 1 was explored in previous results). Error bars are standard errors over 10 random restarts. The entropy approach substantially outperforms these heuristics, demonstrating that it is not only acting through one of these simple mechanisms.

heuristics match the performance of the entropy method, suggesting they are effective when that mechanism is at play. In the independent case, the feature selection performs slightly worse than entropy selection. This demonstrates that, when the data has the corresponding mechanism, the heuristics we designed make sense and work effectively.

5.6.3 Demonstrating that Heuristics are Insufficient in Real Data

Finally, we demonstrate in our real-data examples that these heuristics all underperform the entropy-selection approach. We focus on heuristics 2 and 3 as heuristic 1 was discussed in detail in Section 5.5, including caveats about sometimes needing to first add global features to see good performance. This suggests that the entropy approach is doing something more nuanced than simply applying one of these heuristics, and provides evidence that the task of proposing a good set of features would be challenging for a user without these computationally defined queries.

In both datasets, the entropy approach substantially outperforms the labeling and

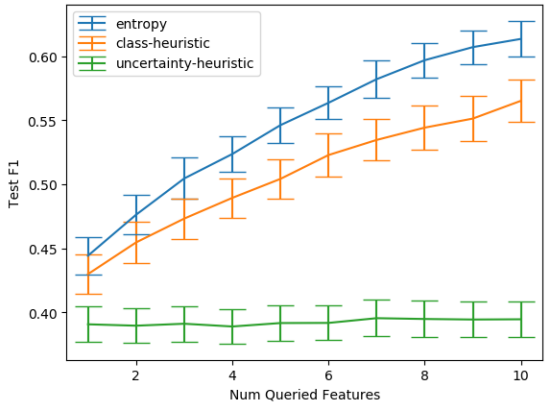


Figure 5.6: Test f1-score as a function of B for both the entropy approach, and all 3 of our proposed heuristics. Error bars are standard errors over 10 random restarts. The entropy approach substantially outperforms heuristics 2 and 3. The caveats about when the entropy approach outperforms the feature-selection baseline in this dataset remain.

feature uncertainty heuristics, suggesting that these heuristics and their corresponding mechanisms are not sufficient to explain the good performance of the entropy approach. Figures 5.5 and 5.6 show the performance of the three heuristics compared to the entropy approach in the recipe and birds datasets respectively. In both of these cases, the entropy approach substantially outperforms the class-heuristic and the uncertainty heuristic. We discuss the feature selection heuristic in detail in Section 5.5. This suggests that neither of these simple heuristics explains the performance gain from the entropy selection approach.

In both datasets, the labeling heuristic performs substantially better than the feature uncertainty heuristic, suggesting that it may be a closer approximation to how the entropy method works. The uncertainty heuristic does not appear to particularly improve the performance of the prediction over the machine-only baseline. However the class-heuristic does improve performance with additional queries, suggesting that while it does not reach the full performance of the entropy method, it may, in part explain its posi-

tive performance.

Summarizing insights While these heuristics do not tell us exactly how a user would choose to select features to provide values for, they do provide some insights that these relatively simple strategies are not sufficient for reaching the performance of the entropy selection method. This suggests that it may be difficult for users to match this performance as well. In addition, even these heuristics require some understanding of the ML model's behavior. These can be communicated to users, but even these simple strategies may not be intuitive to the user.

5.7 Related Work

Human Feature Inputs to ML Predictions Approaches to incorporate human features as inputs to ML predictions exist, but address different problem settings. Some methods use crowdworkers to label features (e.g. [13,82,100,77], however these are employed at train time in conjunction with model development, and assume all features will be labeled at test time. Interactive symptom checkers query patients for symptoms to help them explore diagnoses. Systems including [60] have hard-coded rules about which features to query, rather than learning them. [75] does algorithmically select relevant queries using uncertainty over feature values, which is similar to our approach, but models associations between symptoms and diseases are hard links which does not accommodate general classes of machine learning models. Finally, in geospatial settings, estimation fusion techniques can make predictions combining multiple sensors, some of which can be human-based (see [23,22] for an overview). In contrast, we study a setting where only a small number of human features can be queried at test time, and propose methods that are applicable for arbitrary ML

models on tabular datasets.

Methods for Instance-Wise Feature Selection Various methods have been proposed to select subsets of informative features for specific instances, however none of these specifically address the use case we study. In the active learning vein, we focus on the approach proposed in[65] as an effective solution to the problem we study. Other active learning approaches solve slightly different problems, including[76], and[78]. Other approaches to instance-wise feature selection at test time include:[97] which uses deep learning to discover subsets of predictive features and[10,29], which learns a function to extract a subset of features that are most informative for a given example by maximising the mutual information between some selected features and the response variable. While multiple of these approaches may be effective in this setting, our focus is on defining the problem of feature-level synthesis for human+ML decision making and demonstrating that an effective solution exists, rather than finding the best solution to this problem.

5.8 Evaluation of "Checks"

Some of the checks evaluate by assumption or by design of the method, and some we evaluate experimentally with computational proxies for human input. In future follow up experiments, we aim to evaluate some of these questions further with real human input.

HQ1 and HQ2 we evaluate by assumption. We assume that labeling features for an instance that represents the user is not particularly challenging, as it corresponds to answering questions about their lives like, for example, are they married, or do they exercise at least 3x a week. We also assume that they can accurately answer these questions for the same reason,

evaluating HQ2. There may be questions with less clear-cut answers, and whether and how these fit into this framework is an interesting open question.

HQ3 and HQ4, we do not directly evaluate in this work, however we believe this is an interesting avenue for future human studies to understand how this method works in practice. For HQ4, we do assume that the user will not be able to easily make this prediction themselves as they are, by assumption, not a domain expert who is trained in making the prediction at hand.

MQ1 and MQ4 we evaluate using distinct feature sets in the semi-synthetic experiments we run (where the data is real but the split between machine and human features is synthetic). MQ1 is answered by the 'all features' upper bound, which performs better than the machine-only baseline–this answers question MQ4. MQ2 we evaluate through the end-to-end performance of the system rather than directly. MQ3 we evaluate by comparison to baselines–most importantly feature selection which provides an analogous way of choosing q.

5.9 Discussion and Conclusion

In this work, we define the problem of feature-level synthesis in human+ML decision making and contrast it to existing approaches for combining human and ML insights. We argued that, in contrast to existing approaches for synthesising human and ML insights, this approach provides a way of incorporating insights from users without specific decision-making expertise in the domain, but who may have important insights about instances with which they are associated. We then formalized the problem of feature-level synthesis as querying an instance-specific set–q of informative feature values–h, from the user un-

der a query budget, and demonstrated how an approach introduced by[65] can be used to solve this optimization problem. Finally, we showed in experiments on real datasets with realistic human-machine feature splits that using this approach with an additional heuristic we proposed results in improved performance over not using the human features, or using only a global subset of them. This demonstrates the value and feasibility of this problem of feature-level synthesis in human+ML decision making.

While we do not evaluate how well users could select a set of reasonable feature queries for their own instance, we do propose a set of mechanisms to explain the performance of the entropy method, as well as corresponding heuristics for more simply achieving this performance. We demonstrate that these are not sufficient for achieving the same performance as the entropy approach, suggesting that this strategy is doing something more complicated that may be difficult for a user to match–particularly without a deep understanding of the underlying ML models.

We do, however, make assumptions in this work that, while plausible, merit further study. We propose a method that works given a predictive set of human features, however this may not always be the case. Additionally, even when human features are important, they may be distinct from machine features in ways that impact how they can be used for prediction. For example, important human features such as pain may be more subjective than some machine features. Future approaches could explore ways to identify specific use cases where human features exist and are informative, and methods to account for subjectivity when querying human features. Finally, the assumption that we have access to the human features at train time limits the contexts in which this method can be used. Future work can explore relaxing this assumption by incorporating active learning techniques to

query only those human features that are truly needed to train the model.

In addition to these questions, we see when applying the checks discussed in Chapter 2 that there are additional open questions on the human side. Importantly, these include how well non-expert decision makers can make predictions compared to this approach, and how well users can choose the set of feature values to give for their instance. Having this set of checks can illuminate places like these were human experiments can contextualize the usefulness of this approach.

We demonstrated the importance and feasibility of feature-level synthesis in human+ML decision making as a way of providing non-expert users with effective ways to interact with predictions of ML models that affect them.

6

Discussion, Conclusion and Future Work

In this thesis, we developed a framework for optimizing ML systems to operate in specific sociotechnical contexts based on learning useful proxy functions of important human properties. Our contributions include a formal description of the framework, a set of guidelines for developing methods under this framework, and three case studies of methods that instantiate this framework for different choices of human properties.

6.0.1 Discussion

We first summarize insights derived from applying this framework and the evaluation template to the 3 methods discussed in this thesis. These insights fall into the categories of reflecting on our proposed "checks;" considering at what level of specificity to define goals; and considering different categories of human proxies that are useful for evaluating the checks.

Reflecting on the goal of using learned proxies to optimize for sociotechnical context The goal we put forth at the beginning of this thesis is to develop methods that allow us to optimize machine-learning models to work better in socio-technical systems. Does this framework of learning proxies for human functions allow us to do this? In the three works we discuss in this thesis, this framework does not fully solve the problem of how to optimize ML models for socio-technical systems, however it does provide one useful tool for doing so given a specific abstraction of an interaction between a user and an ML model. ML models can play different roles in decision-making including as an aid to both patients and clinicians, or as a way of counteracting resource constraints[28]. Deciding on the role of the ML model is an important aspect of this–one we operationalize through the choice of the human function h. Once this has been decided, our framework allows us to optimize models based on a reasonable approximation of h. However, the choice of h–i.e. how users and ML models interact, is challenging. One way to approach this is through careful work with users to understand their needs from ML systems such as the work described in Jacobs et al. [28]. We hope to incorporate this style of work into choosing appropriate human functions in future projects.

Reflecting on proposed "checks" In analyzing if and how we evaluated each of the proposed checks for our framework, we can pull out common themes, best practices, and areas for improvement. The three trends we discuss are the challenges in completing the human evaluation final step, the choice to address some checks by assumption, and the choice to evaluate systems end-to-end rather than addressing each check individually.

The first trend across these 3 works is the difficulty in getting to the final stage of human evaluation. In Chapter 3, we ran a user study to test our method and found some unexpected surprises, such as the high variance of human responses to the task. In the other two systems, we leave the full testing of the system for future work. Our proposed template leaves answering the human questions with real human data as the final step in evaluation, which means that step is also the most likely to be left to future iterations of a project. It also means that by the time we run this evaluation, it is late to make any substantive changes to the method. Evaluating the system with reasonable proxies for human responses addresses some of these concerns, but will not address cases where both our proxies and our method overlook important aspects of a human function, such as the surprising response variance in Chapter 3.

The second trend is the evaluation of some questions–particularly related to the human experience of the task, by assumption rather than experimentally. HQ1 and HQ2–related to the ease of completion of the task for the user, and the accuracy of their response, we generally settled by assumption based on how we designed the specific task. For example, we assumed in Chapter!4 that providing feedback on a specific feature-concept pair, would not be too difficult, however providing a full concept definition by hand likely would be. These two cases are extreme examples where our assumption about their relative difficulty

is likely correct without needing to test it experimentally, but there are also potential tasks with much more nuanced differences between them wrt to HQ1 and HQ2. Evaluating these differences experimentally may help us make more nuanced choices about task design.

The third trend relates to the choice to evaluate the system end-to-end rather than to evaluate each sub-piece of the questions separately. For example, in Chapter 4, we evaluate the full system performance, convolving the learnability of the function and the quality of the query set. While this works fine when both work, it does not provide feedback about exactly which part is breaking in cases where it is not working yet. These questions can be used to guide that inquiry, however in cases where a full system works, that may not be necessary.

Defining goals A second key insight after contextualizing the three works in this thesis under our framework is the question of at which level of specificity to define the goal. The main argument in our framework is that we can more effectively incorporate human insights into the training and deployment of ML systems with *learned* computational proxies for human feedback, rather than hand-derived proxies. However this moves the hand-defining work from defining the proxy itself, to defining a task that can be used to learn the proxy. In some cases, the task and goal may be relatively clear-cut, for example in Chapter 5, where the task is to fill in missing feature values and the goal is to improve predictive performance. In other cases, the link between the task and the overall goal of incorporating human insights is more tenuous. For example, in Chapter 4 where the goal we stated is to learn a set human-aligned concepts. However there is a higher-level goal implied, which is to allow users to make more effective decisions with the input of ML systems. There is also the lower-level goal of associating features and concepts correctly. Our task definition is

clearly linked to the lower level goal, but as we zoom out and define the goal more broadly, untested assumptions come in. How to link high-level goals with specific, actionable tasks that preserve those properties, but efficiently address a subpart of learning or operating an ML system is an interesting open question.

Human proxies for testing. Finally, we notice in our discussion of results that there are two kinds of hand-derived human proxies we use in these results, and reflect on their importance to answering different questions. The first kind of proxy is generalizable–i.e. computable for a new dataset without new hand-engineering. This allows it to be used in optimization without requiring human feedback. Examples include correlation between features, mean path length of a decision tree, etc. The other kind of proxy is hand-derived for a specific data–for example the hand-derived concepts in Chapter 4. These cannot be used in optimization for a new dataset without re-hand-deriving them, however they can be useful in allowing us to upper bound performance, answering questions about the usefulness of the defined human function without convolving it with our ability to learn a model of the function, or select a useful query set. More clearly specifying which of these types of proxies is most appropriate for answering the questions in each of the checks is future work that will provide further insights into applying those checks in practice.

6.0.2 Future Work

In addition to these insights, there are many concrete avenues for future work. We focus on two main aspects: 1) broadening the set of considerations in the "checks" we propose; 2) other "template" approaches for balancing the cost of evaluating the "checks".

Validating the use of learned proxies against optimizing models directly with humans. In this work, we argue that proxy functions are a valuable way to save human time in function-evaluation-intensive ML optimization frameworks. We introduce tools for learning proxy functions from real human responses to incorporate human feedback into these data-intensive optimizations. While we demonstrate through simulated results and some user studies that these can extrapolate reasonable approximations of what true human functions h look like in practice, further studies can further explore the claim that more queries to a lower quality approximate human function during optimization are more useful than few queries to an exact human function.

Broadening criteria for success. The checks we provide to evaluate methods focus on performance, but there are other important aspects of decision-making like fairness and human agency. For example, in Chapter 3, the human function evaluation requires users to answer questions that are not transparently related to the model parameters limiting the agency the user has in this process. In contrast, in Chapter 4, the human function evaluation allows users to give direct feedback on the model parameters in a transparent learning process. While this gives the user more agency over the optimization, this desirable property is not formalized in the "checks" we propose.

Future work could expand the checks to include other dimensions of human-ML interactions when considering the success of a particular problem formulation. This could help prioritize the development of methods that not only produce accurate decisions, but also consider qualitative aspects of how users interact with the system.

Considering other template approaches. The "template" we describe provides one way to approach the two problems of the expense of evaluating many of the checks, and the dependence of the human and machine checks. However there are other possible approaches which may have distinct advantages and disadvantages. Our approach focuses on spending the method development cost before the user evaluation cost by developing methods with human proxies. However this has the disadvantage that one could develop methods that work for proxy human functions, but not the real thing. In Chapter 3, we developed a method without considering the noisiness of the human function. When we evaluated the method with users, we found that the variance of the function estimates were higher than expected because of this. While the method works anyways, and would simply require the collection of more data to reduce the variance, this may not always be the case.

One could alternatively approach the problem with the opposite strategy by first running user studies to evaluate the human properties, then using those results in development. In this case, it would be useful to have a way of guessing whether a specific human property would lend itself to method development in an analogous way to how we use human proxies to guess whether a method will work with human input. Which of these approaches is more appropriate for developing methods under this framework, and in which contexts, is an open question.

6.0.3 Summary

In summary, methods that learn an approximation to important, missing human properties can be used to optimize ML systems to operate in sociotechnical contexts without the cost of constantly querying users for the property. In at least some contexts, they also

improve over hand-specified proxies that are not learned using human-property data. We present a framework for formalizing methods of this type, and guidelines for developing new methods under this framework.

A

Method 1: Optimizing for the Human Interpretability Function

A.1 Similarity Kernel for Models and GP parameters

Model-based optimization requires as input a notion of similarity. We use an RBF kernel between feature importances for decision trees, and between a gradient-based notion of fea-

ture importance for neural networks (average magnitude of the normalized input gradients for each class logit).

We use the scikit-learn implementation of Gaussian processes[59]. We set it to normalize y automatically, restart the optimizer 10 times, and add $\alpha = 10^{-7}$ to the diagonal of the kernal at fitting to mitigate numerical issues. We used the default settings for all other hyperparameters, including the RBF kernel (on the model features above) for the covariance function.

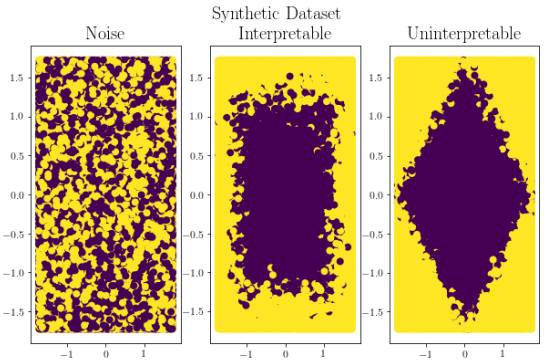


Figure A.1: We build a synthetic data set with two noise dimensions, two dimensions that enable a lower-accuracy, interpretable explanation, and two dimensions that enable higher-accuracy, less interpretable explanation. The purple data points are positive and the yellow are negative. Data points were generated for each set of two features independently, then points sharing the same label in all dimensions were randomly concatenated to form the final dataset.

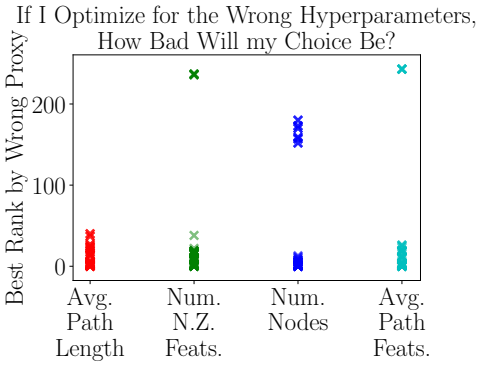
A.2 Experimental Details: Identifying a Collection of Predictive Models

We train decision trees for the synthetic, mushroom and census datasets with a test accuracy thresholds of 0.9, 0.95 and 0.8 respectively. On the synthetic dataset, 0.9 is slightly higher than the accuracy we can achieve on the interpretable dimensions. We make this choice to avoid learning the same, simple model over and over again. On the mushroom dataset, we can achieve a validate accuracy of 1 with decision trees, and on the Census

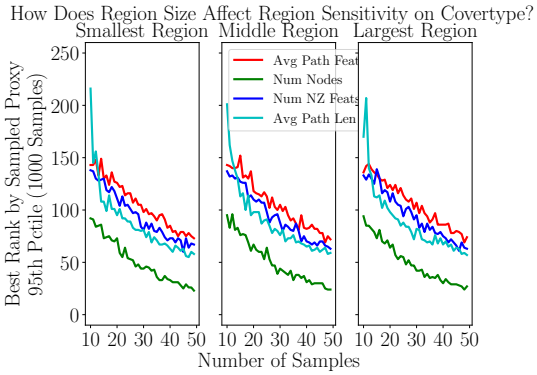
dataset, we can achieve a validate accuracy of 0.83 with decision trees. In both cases, we set the accuracy thresholds slightly below these numbers to ensure that we can generate distinct models that meet the accuracy threshold. For each of these, we train 500 models.

To produce a variety of high-performing decision trees, we randomly sample the following hyperparameters: max depth [1-7], minimum number of samples at a leaf [1, 10, 100], max features used in a split [2 - `num_features`], and splitting strategy [best, random]. The first two hyperparameters are chosen to encourage simple solutions, while the last two hyperparameters are chosen to increase the diversity of discovered trees. We use the scikit-learn implementation [59], of decision trees and perform a post-processing step that removes leaf nodes iteratively when it does not decrease accuracy on the validation set (as in Wu et al. [96]).

We train neural networks for the covertype dataset with an accuracy threshold of 0.75. We can achieve an accuracy of 0.71 with logistic regression, so we set the threshold slightly above that to justify the use of more complex neural networks. For the neural network models, we randomly sample the following hyperparameters: L1 weight penalty [0, 0.0001, 0.001, 0.01], L2 weight penalty [0, 0.0001, 0.001, 0.01], L1 gradient regularization [0, 0.01], activation function [relu, tanh], architectures [three 100-node layers, two 100-node layers, one 100-node layer, one 25-node layer, one 250-node layer]. These are then jointly trained according to the procedure in Ross et al. [73] for 50 epochs for batch size 512 with Adam. (We train between 1 and 4 models simultaneously, another randomly sampled hyperparameter). For this dataset, we train 250 models.



(a) We found the best model by each proxy for every setting of the region hyperparameters, and computed its rank by the same proxy for every other setting of the region hyperparameters. Each $\times$ corresponds to one of these pairs. The highest values all correspond to the variance scaling factor 0.1. The other two settings of this hyperparameter tend to agree on how to rank neural networks.



(b) We found the best model(s) by each proxy and computed their rank by the same proxy computed on a sample of data points. The comparable values of the lines across all three plots indicate that we need a similar number of samples to robustly rank neural networks for the smallest, middle and largest region settings (we do not include cross pairs).

Figure A.2: Neural network local explanation sensitivity analysis

A.3 Experimental Details: Parameters and Sensitivity to Local Region Choices

We can ask humans to perform the simulation task directly using decision trees, but for the neural networks, we must train simple, local models as explanations (we use local decision trees). This procedure requires first sampling a local dataset for each point x we explain. We modify the procedure in Ribeiro et al. [69] to sample $10,000$ points x' in a radius around the point x defined by its 20 nearest neighbors by Euclidean distance. We then binarize their predictions $M(x')$ to whether they match $M(x)$ and subsample the more common class to balance the labels. We do not fit explanations for points x where the original sampled points x' have a class imbalance greater than 0.75; we consider these points not on the boundary. Finally, we return the simplest tree we train on this local dataset with accuracy above a threshold on a validate set. We randomly set aside 20% of the sampled points for

validation, and use the rest for training. (Note: if we were provided local regions by domain experts, we could use those.)

Our procedure for sampling points around some input x uses two hyperparameters: a scaling factor for the empirical variance, and a mixing weight for the uniform distribution for categorical features that we use to adjust the empirical distribution of the point's 20 nearest neighbors. We use 0.01 to weight the variance and 0.05 to weight the categorical distributions. Finally, when training the trees, we set a local fidelity accuracy threshold of 90% on a validation set and iteratively fit trees with larger maximum depth (up to depth 10) until one achieves this threshold. (We assume data points with local models deeper than this will not be interpretable, so fitting deeper trees will not improve our search for the most interpretable model.) We require at least 5 samples at each leaf. We use the scikit-learn implementation [59] to learn the trees and perform a post-processing step that removes leaf nodes iteratively when it does not decrease accuracy on the validation set (as in Wu et al. [96]).

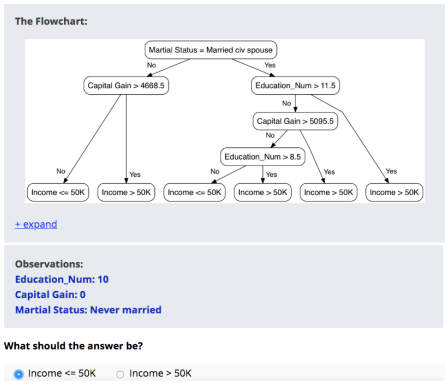
How sensitive are the results to these choices? In Figure A.2a, we first identify which of our $K = 250$ models would be preferred by each interpretability proxy if the local regions were determined by variance parameters set to [0.001, 0.01, 0.1] and the mixing weights set to [0.01, 0.05, 0.1] (9 combinations). Next, for each of those 9 models, we identify what *rank* it would have had among the K models if one of the other variance or weight parameters had been used. Thus, a rank of 0 indicates that the model identified as the best by one parameter setting is the same as the best model under the second setting; more generally a rank of r indicates that the best model by one parameter setting is the r-best model under the second setting. The generally low ranks in the figure indicate agreement amongst the different choices for local parameter settings. The highest mismatch values for the number

of nodes proxy all correspond to the variance scaling factor 0.1 (which we do not use).

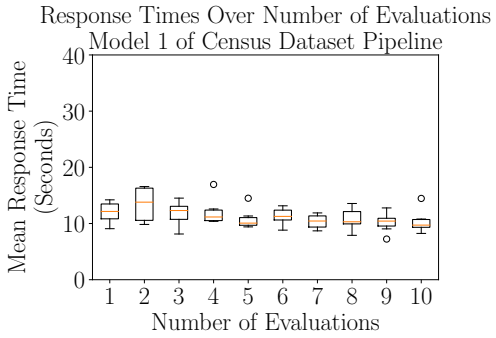
Do we need more points to estimate model rank correctly for any of these region settings? We find the best model(s) by each proxy, then we re-rank models using a small sample of points to compute the same proxy. We do this for the smallest, middle and largest settings of the local region parameters (we do not include cross-pairs of parameters in these results). Figure A.2b shows that different hyperparameter settings require similar numbers of input samples x to robustly approximate the integral for $p(\mathcal{M})$ in equation 3.4 for a variety of interpretability proxies substituted for HIS.

A.4 Experimental Details: Human Subject Experiments

In our experiments, we needed to sample input points x to approximate the prior $p(\mathcal{M})$ in equations 3.2 and 3.4. For globally interpretable models, we ask users about the same data points across all models to reduce variance. In the locally interpretable case, we would only conduct user studies for points near the boundary (in $B(\mathcal{M})$) and would thus sample points specific to each model's boundary. Each quiz contained 8 or 16 questions per model (8 for the pipeline experiments, 16 for the Amazon Turk experiments), with the order randomized across participants. There was also an initial set of 3 practice questions. If the participant answered these correctly, we allowed them to move directly to the quiz. If they did not, we gave them an additional set of 3 practice questions. We excluded people who answered fewer than 3 of each set of practice questions correctly from the Amazon Mechanical Turk experiments.



(a) An example of our interface with a tree trained on the census dataset with the fewest non-zero features. In our experiments, we show people a decision tree explanation and a data point including only the features that appear in the tree. We then ask them to simulate the prediction according to the explanation.



(b) We asked a single user to take the same quiz 10 times to measure the effect of repetition on response time. The difference in mean response time between the first and last quiz is around 2 seconds. The y-axis scale is the same as that in 3.4a so the magnitude of the learning effect can be directly compared to the magnitude of the differences between models in our experiment.

Figure A.3: Interface and learning effect

Experiments with Machine Learning Graduate Students and Postdocs

For the full pipeline experiment, models were chosen sequentially based on the subjects' responses. We collected responses from 7 subjects for each model in the experiment with the census dataset, and from 9 subjects for each model with the mushroom dataset.* Accuracies were all above 85%. We ran 10 iterations of the algorithm, each a quiz consisting of 8 questions about one model, and two evaluations at the end of the same format. We used the mean response time across users to determine $p(M)$. We did not exclude responses, and participants were compensated for their participation. Using the same set of subjects across all of these experiments substantially reduced response variance, although the smaller total number of subjects means we did not see statistically significant differences in our results.

*We recorded 2 extra responses for iteration number 4, and 2 fewer responses for iteration number 5 in the census experiment, and 1 extra response in iteration number 3 for the mushroom experiment due to a technical error discovered after the experiment, but we do not believe these affected our overall results. (Extra responses are from the same set of participants.)

Experiments with Amazon Mechanical Turk We had initially hoped to use Amazon Mechanical Turk for our interpretability experiments. Here, we were forced to use a between-subjects design (unlike above), because it would be challenging to repeatedly contact previous participants to take additional quizzes as we chose models to evaluate based on the acquisition function.

In pilot studies, we collected 33 and 24 responses for the two models selected by the pipeline (the first had a medium mean path length, and the second had a high mean path length), after excluding people who did not get one of the two sets of practice questions right, or who took less than 5 seconds or more than 5 minutes for any of the questions on the quiz. The majority of respondents were between 18 and 34. We asked participants 16 questions with a 30 second break halfway through. We paid them $2 for completing the quiz.

The first model, which had a medium mean path length, had a mean time of 31.62s (28.61s - 34.63s), and a median time of 26.86s (23.09s - 30.63s) (standard error and median standard error in parentheses respectively). The second model with a high mean path length had a mean response time of 30.94s (28.66s - 33.22s), and a median response time of 30.32s (27.47s - 33.17s). These intervals are clearly overlapping. We could gather more samples to reduce the variance, but cost grows quickly; running one experiment with the end-to-end pipeline with these sample sizes would have cost around $1,000.

B

Method 2: Optimizing Models for Human-Concept-Alignment

B.1 Dataset

B.1.1 Toy data

We generated the toy dataset with 10,000 instances and split it 60%/20%/20% train, valid, test. The pairs of correlated features were generated by first sampling a binary random variable $\sim$ Bernoulli(0.25), then for each of the 2 correlated features, flipping the variable with p=0.05. Labels were generated using the model described in Figure 4.2

B.1.2 Real datasets

We used the Yelp dataset* from the Yelp dataset challenge. To process the data, we kept restaurants with at least 5 reviews, and used a bag of words feature representation, counting the number of times each word appears in all associated reviews for a restaurant. We then labeled as positive examples restaurants with star ratings ≥ 4, and as negative examples restaurants with star ratings ≤ 2, and subsampled the positive class to generate a class-balanced dataset. The words that we kept in the feature vectors occurred in reviews for between 10% and 25% of restaurants, allowing us to find features that were common enough

*https://www.yelp.com/dataset

to be useful predictors, but not so common that they were used for most restaurants.

We used a dataset of patients from 2 New England hospitals with at least 1 MDD diagnosis (ICD9 codes 296.2x, 296.3x) or depressive disorder not otherwise specified (311), and without codes for schizophrenia, bipolar, and typical antipsychotics. Our prediction task was to determine whether the patient will be prescribed an atypical antipsychotic (Olanzapine, Quetiapine, Risperidone, Lurasidone, Aripiprazole, Brexpiprazole, Ziprasidone) within the year after their index antidepressant prescription. We subsampled negative examples to class balance the dataset. Feature vectors consist of counts of how often each ICD9, procedure and medication code are recorded for the patient in the 2 years preceding the index antidepressant prescription. We exclude codes that occur for less than 1% of patients since there is a long tail of these codes that will not be highly predictive since they are recorded for few patients. We additionally remove numerical features from the dataset (patient age and date), and gender markers. We do this so we can use gender as a concept in our simulation studies that must be defined through proxies rather than through the recorded marker. In a real, clinical application, these features would be included in the dataset.

Ground Truth Concept Definitions. We defined a set of concepts and ground truth features associated with each concept. The concepts and features were generated by the first author, including input from domain experts in the Psych domain. Concepts were chosen to be reasonably predictive, and seed features were selected to be representative of the concept (based on the judgment of the first author, who compiled these lists of features associated with each concept). At each restart, concepts are seeded with one bolded term from each concept, such that f generated with the seed features had large enough concept weights for each concept that adding new features to the concept definition affected the

accuracy (this was sometimes not the case when a concept was seeded with a term that was irrelevant to the prediction task). Several features were excluded when they frequently lead to this condition being violated, suggesting that these were not predictive seed features. In practice, one would validate the predictiveness of the concepts and seed features before running this approach, which our choice of seed features mimics.

The concepts we define in the Yelp dataset are: 'mention of service', 'positive ambiance', and 'positive food texture'. The associated words for each concept are listed below, with potential seed features in bold (concepts are seeded with a randomly chosen one of these):

'mention of service': management, manager, server, waiter, waitress, employee, hostess, cashier, bartender, orders, ordering, servers, register, refill, serves, serve, waitresses, refills, refill, **managers**, **reservation**, reservations, services, waiters

'positive ambiance': **cozy**, **ambience**, **ambiance**, welcoming, casual, friendly, music, modern, neighborhood, atmosphere, **comfortable**, quaint, **vibe**, comfort, comfortable, mood, welcome

'positive food texture': tender, **crispy**, crisp, **juicy**, **creamy**, moist, crunchy, **fluffy**, crunch

The concepts we define for the Psych dataset are: 'anxiety' , 'gender-female' , 'hospital-ed' (hospitalization/emergency department visit) , 'addiction'. The associated featuress for each concept are listed below, with potential seed features in bold:

'anxiety': **30002**, **30000**, 30001, 7992, 3003, **rxnorm:2598**, **rxnorm:596**, rxnorm:6470, rxnorm:2353, rxnorm:3322, rxnorm:7781

'gender-female': v242, **c76801**, c59051, c58100, c76830, c76815, c76816, 6260, rxnorm:214559, v7610, 7210, 650, c76819, rxnorm:6691, c88142, c88141, v221, c59409, 6271, p7569, 6262, 64893, 6264, v103, 2189, p7534, c76805, v222, v7611, 6160, c59400, **c81025**,

c82105, c76645, rxnorm:4100, 61610, v7231, v270, c76811, v163, rxnorm:214558, c88174, **drg:373**, 6202, rxnorm:384410, rxnorm:6373, **c59025**, 6253, c88175, 1749, 6221, 6259, 6268, 6272, 6289, 79380, 7950, 79500, c76090, c76091, c76092, c77057, **v7612**, **v762**, c82670, 65963, rxnorm:324044, c84146, v220, rxnorm:4083, c76817

'hospital-ed': **c99232**, c99231, c99222, c99233, c99238, c99223, c99282, c99285, **c99284**, c99283, c99281, c99239, c99253, c99219, c99218, **c99221**, **zINPATIENT**, c99254, c99252

'addiction': 30400, **c80100**, **3051**, **30500**, 29181, 30390, 30590, c82055, 30490, rxnorm:6813, **rxnorm:7407**, c80101, v1582

Codes starting with 'c' are CPT codes, codes starting with 'rxnorm' are medication codes, and the rest are ICD9 codes.

B.2 Hyperparameters

B.2.1 Our Approach

We train f using the scikit-learn implementation of logistic regression with no regularization term [58].

B.2.2 Interactive Concept-Learning Baselines

For both interactive concept-learning baselines, we train the same f (unregularized logistic regression implemented in scikit-learn [58]) as we do for our method. We use the concept probabilities directly in the downstream classification. This gives these approaches an advantage over our model and makes them slightly less interpretable, since our concepts are always constrained to be in $\{0, 1\}$.

Concept Classifiers Inspired by Amershi et al. [3], we tune a set of concept-classifiers using concept labels, where the classifiers are l1-penalized logistic regressions so as to be simulatable. We request labels for the example that most improves the downstream accuracy of the model after retraining from a random subset of examples (while we use ground-truth concept labels in our simulation experiments, these would need to be estimated in practice). We search over a random subset of 100 examples to consider labeling. While searching over more examples will likely improve performance of the approach, it also increases the running time, which can seriously impact user experience in an interactive system. We generate the initial set of labels for each concept by labeling as positive examples of the concept all examples that have the seed term for the concept and randomly choosing 1 negative example of the concept to label. This gives the approach a roughly equivalent starting amount of information to our approach, which requires a seed term.

To determine the weight of the l1 regularization, we do grid search over a set of values to find the weight that produces final concept models with approximately the same number of features as our method has features associated with any concept. In practice, we search over values in the range 0.01 to 0.1 (after predetermining that the weight lies in this range), taking steps of size 0.01. We z-score the features before using this approach.

Topic Model We also compare to the method in Lund et al. [47] that tunes supervised topics through a set of curated anchor words to use in downstream prediction tasks. To make the interactions comparable to our approach, we propose a new anchor word as the highest probability word for the topic that is not already an anchor word, or a downstream label. We add rejected proposals to a set of "irrelevant concepts" not used in prediction since topic models must model all of the data–a feature not shared by our approach.

We run two variants of the topic-model approach in our experiments that create the "irrelevant topics" in two ways: in the first variant, we seed the model with 5 times as many non-concept-related topics as concept-related topics. We start with 1 topic for each concept, and each time we reject a term, we create a new topic with that word as the anchor word. Before adding rejected features to a new concept, we verify that they do not belong to the lists of related features for any other concepts. If they do, we ignore them since we do not want to prevent them from being suggested for the correct topic (although this would not be doable in practice). We call this TM, and report these results in the main body of the paper; they are better than TM-2.

In the 2nd variant, TM-2, we generate anchor words for these by, for each new topic, taking the word that is the furthest from the existing anchor words using the Jaccard distance metric. We then assign rejected words to these topics by taking the topic with the closest anchor word to the rejected word based on Jaccard distance. These two variations allow us to explore whether pre-seeding the model with these "irrelevant topics" and allowing it to learn topics that more accurately correspond to our desired concepts from the beginning, or if creating "irrelevant topics" to specifically capture things that may be confused with our desired concepts is more effective. For both variants, we use $5 * C$ irrelevant topics.

We infer the topics by drawing a small number samples (specifically 10) of the topic vectors as suggested in Lund et al.[46] and computing probabilities by normalizing. We then binarize these to compute concept accuracy by taking all topics where the probability is greater than 0.1 for the document as 1 and the other topics as 0. Note that we train the logistic regression model for downstream prediction only on the topics that correspond to our desired concepts.

B.2.3 Non-Interactive Baselines

The random forest model has 200 estimators, and we tune the maximum depth of the trees over the range $[5, 10, 25, 50, 100, \text{None}]$ using 5-fold cross validation. The neural network has 1-hidden layer with the same number of hidden nodes as our approach has concepts– this is the comparable architecture to our approach. We use a sigmoid activation function and ADAM as an optimizer, and search over step sizes from $[0.001, 0.01, 0.1, 1.]$ using 5-fold cross-validation. We use batch size 32 and run it for 1000 iterations. For LR, we choose the l1-penalty that produces the closest number of non-zero coefficients to C (the number of concepts). We search over values in the range 0.0001 to 1., taking steps of size 0.0001 between 0.0001 and 0.001, of size 0.001 between 0.001 and 0.01, etc.

We z-score the features before using any of these approaches. We trained the random forest model, and the logistic regression models using the scikit-learn implementations[58].

B.3 Results

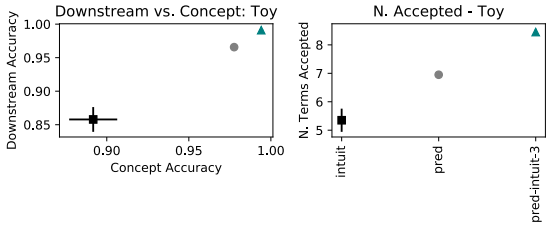


Figure B.1: Downstream vs. concept accuracy on the test set for our proposed variant, the pred variant, and the intuit variant, as well as the number of accepted proposals made by each variant for 20 random restarts. Our variant works best by all metrics, followed by pred, then intuit.

Toy Example: Our approach achieves perfect downstream and concept accuracy while pred and intuit do not. Figure B.1 shows the downstream accuracy plotted against

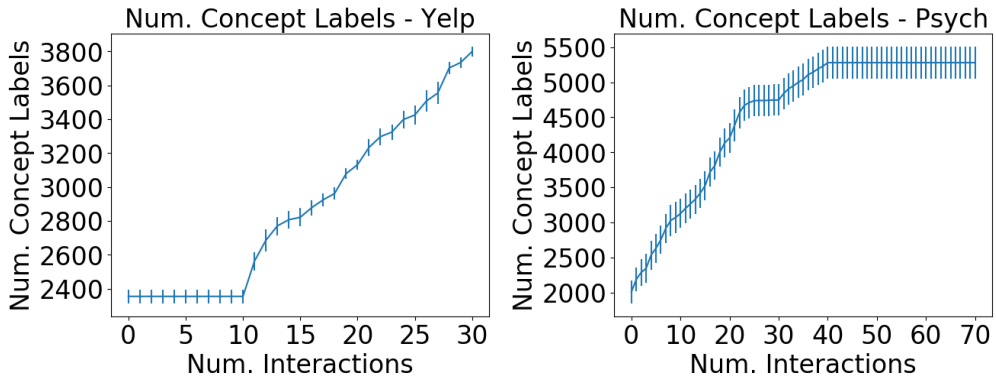


Figure B.2: Number of positive concept labels in test set by number of proposals for our method. Yelp on left, psych on right. The number of positive concept labels substantially increases suggesting our method expands coverage.

	Yelp		Psych	
Variant	Downstream	Concept	Downstream	Concept
TM-2	0.71±0.01	0.61±0.01	0.61±0.00	0.64±0.00

Table B.1: Downstream accuracy and concept accuracy ± standard errors for the TM-2 variation of the anchor topic modeling baseline. These results are worse than the TM variant reported in the main text.

the concept accuracy as well as the number of accepted features for the pred, intuit and our proposed variant in the toy example. Our method works best, achieving perfect downstream and concept accuracy in 4 proposals per concept, followed by pred, then intuit. Each random restart corresponds to re-sampling the dataset and re-running the algorithms.

Inclusion and Coverage: Our approach increases positive concept labels substantially, implying improved coverage. Figure B.2 shows the number of positive concept labels by number of user interactions from the experiment above. In both domains, the number of positive concept labels grows substantially, and more than doubles from the seed features in the Psych domain. This has implications for fairness and robustness as it allows for multiple synonymous ways of coding for different concepts that capture different instances. In our model, these can all be recognized instead of relying on a single common

coding as would likely be the case in a model constrained only to be sparse (without concepts, like LR).

The TM-2 variant has worse performance than TM. Results for the second variation of TM are in Table B.1. Across the board, these are the same or worse than the TM results reported in Table 4.1, so we only compare TM to our approach.

C

Method 3: Optimizing Elicitation of Human Context at Test-Time

C.1 Real Data Experiments

C.1.1 Hyperparameters

We implement all logistic regression models using the scikit-learn package[59]. We used a multinomial loss, the SAGA solver and we set the maximum number of iterations to 5000. In our hyperparameter search for the real data experiments, we searched over penalties: [l1, l2 and none], inverse regularization strengths [0.01 , 0.1 , 1. , 10. , 100.], and class weighting schemes [none, balanced]. We set $B = 10$. For the entropy selection approach, we set the number of Monte Carlo samples $S = 5000$ so as to minimize this source of approximation error. For the models used in the entropy method to model f and $p(x^b|x^m)$, we do not search over balanced class weights as this causes challenges with calibration– something that is important for this method because it depends on accurately estimating probabilities.

We perform hyperparameter selection based on maximizing f1-score (the metric we report) on the validation set, except in the case of the models for $p(x^b|x^m)$, where log-loss is minimized instead. This is to encourage those models to accurately model the probability distribution they approximate. For the feature selection models, we chose the hyperparam-

eters once based on the machine features rather than re-optimizing them for each possible choice of features to select.

C.1.2 Data Preprocessing

We preprocess the recipe dataset by removing instances without labels, then subsampling 15% of the remaining instances, maintaining overall label distribution, for computational reasons. We then removed features with positive values for less than 100 instances. For the birds dataset, we use the coarse class labels, and remove instances with infrequent class labels, recorded for less than 50 instances. We then binarize features values at 0.5 (features are recorded as the fraction of agreement on MTurk responses about whether the feature is present in the image), and removed features with positive values for less than 100 instances. We split the data, class-balancing labels, into train/validation/test sets of sizes $\frac{2}{3}$, $\frac{1}{6}$, $\frac{1}{6}$ of the instances respectively.

C.1.3 Additional Results

In the birds dataset, a follow-up experiment shows that doing a round of global feature selection, then running the entropy-selection method can increase the performance of this method relative to the feature selection baseline. Figure C.1 shows the results of a follow-up experiment where we added the first 6 features selected by *feature-selection* to the machine feature set for the birds data, and re-ran all 3 approaches for 10 additional queries. This includes only 5 random restarts. Here, we see that *entropy-selection* and *entropy-refit* perform similarly for the first 4-5 queries, then substantially outperform *feature-selection* with an f1-score of 0.69 vs. 0.65.

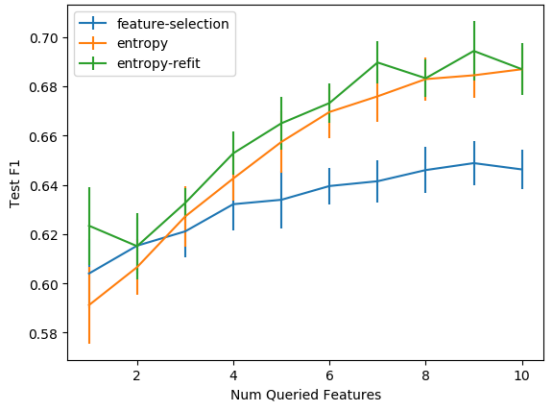


Figure C.1: Test f1-score on *birds* as a function of B after adding first 6 *feature-selection* queries to machine features and re-running. Error bars are standard errors from 5 random restarts. By query 6, both entropy methods substantially outperform *feature-selection*.

This suggests several things: 1) the re-training heuristic is useful for boosting performance in at least some cases, 2) even without the retraining heuristic, *entropy-selection* is useful, at least eventually, and 3) for some datasets, individualized feature queries are useful immediately, while for others, they may be more useful after querying an initial set of shared features.

The late improvement of the entropy based methods over *feature selection* in *birds* suggest there may be cases where a combination of instance-specific and global human feature queries can be useful. Our proposed *entropy-retrain* method still performs as well as *feature-selection*, but there may be benefits to recording the same features for all instances, including using this information to update $\hat{p}(x^b|x^m)$. Future work could explore heuristics for identifying when a dataset would benefit from an initial round of global human feature collection, and when it would directly benefit from an instance-specific approach.

C.2 Toy Example

We demonstrate the entropy selection approach on a pedagogical toy example where important human features with high uncertainty can be predicted from the machine features. If the important human features were fully predictable from the machine features, they would be unnecessary for classification, and if they were not predictable at all, our approach would not work.

C.2.1 Data generation

We generate a dataset with 10 human features, 20 machine features, and 5000 instances where each of the human features has a corresponding, unique machine feature that, when activated, corresponds to the value of the human feature being highly uncertain. In all other cases, the human features are highly unlikely to be activated for an instance. Exactly one of these machine features is activated for each instance. The remaining 10 machine features are uncorrelated with the human features, and are activated with $p = 0.5$. Labels are then sampled from a linear regression model with randomly generated weights where the paired human and machine features have relatively higher weights than the other machine features, as they are sparser.

Generating X^h We construct $p(x^h|x^m)$ so the machine features tell us which human features may be positive without giving us too much information about x_h. For each dimension of x_h, we set the weights of the corresponding logistic regression model to predict $p(x^h_d|x^m)$ so that exactly 1 dimension, d' randomly sampled from $\in \{1, ..., D_m\}$, has weight $4.5 + \varepsilon$, and all other dimensions have weight $\varepsilon \sim N(0, 0.01)$. We set the bias term

$w_d^0 = -4.5 + \varepsilon$. This has the effect that if $x_{d'}^m = 1, p(x_d^h = 1$ is approximately 0.5, and otherwise it is low. We construct the probability distribution for each dimension of X_d^h in this way, sampling d' without replacement. To generate the dataset, we sample X^h from this distribution, after first sampling X^m as described below.

GENERATING X^m We define a sampling procedure for X^m so that each instance has a single human feature that may be positive (and those should be different for different instances). We achieve this by setting exactly 1 of the sampled d' used in the previous step to be set to 1 for each instance. For the remaining features not in the set of sampled d', we sample them with $p = 0.5$. Since each human feature has a distinct d' that signals it may be positive, sampling exactly one of these guarantees that the probability of the human features being on will only be high (around 50%) for 1 human feature.

GENERATING Y Finally, to generate Y from the sampled X, we generate the weights of a logistic regression model as follows, then sample Y from the probability distribution $p(Y|X)$ defined by this model. We do this so the human features are important when they are positive. To achieve this, we set each weight for the human features to either $3 + \varepsilon'$, or -3 $+ \varepsilon'$ with equal probability (this makes it so sometimes the human features make the label more likely to be positive, and sometimes more likely to be negative) where $\varepsilon' \sim N(0, 0.1)$. We also set the weights of the machine feature indices selected in step 1 as d' in the same way. For the remaining dimensions of X^m, we sample weights at either $1 + \varepsilon$ or $-1 + \varepsilon$ so they still influence the prediction, but to a lesser degree. This is to balance the fact that each instance is likely to have multiple of these dimensions that are positive vs. a single one of the other 2 sets of dimensions, and we don't want any one set to dominate the prediction. We

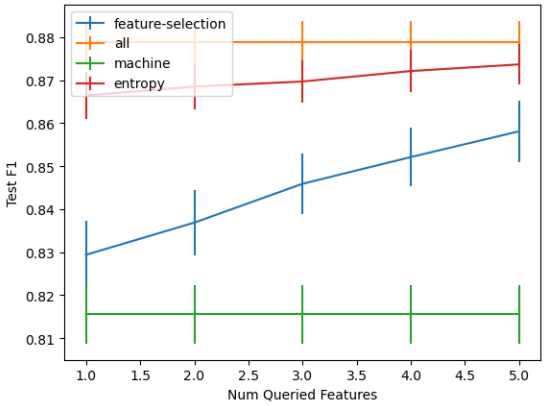


Figure C.2: Test f1-score as a function of B for our propsed method, as well as baselines and upper bound. As expected, *all-features* performs best with f1-score around 0.88 and *machine-only* performs worst with f1-score around 0.815. *entropy-selection* performs almost as well as the upper bound from the first query with f1-score around 0.87, but *feature-selection* only begins to approach this performance after all 5 queries, with f1-score around 0.86.

set the bias to 0. We repeat this procedure until the labels are approximately class balanced, with mean between 0.4 and 0.6.

C.2.2 Experiment

Hyperparameters We split the data, class-balancing labels, into train/validation/test set of sizes $\frac{2}{3}, \frac{1}{6}, \frac{1}{6}$ of the instances respectively. We set the hyperparameters to penalty: none, class weighting: none, and $B = 5$. Aside from these, we use identical hyperparameter settings to those described in the main paper.

Results: Our approach reaches near-optimal performance quickly, while standard feature selection improves much more slowly. Figure C.2 shows the results, averaged over 10 random restarts, of our method compared to feature selection, the full set of features, and the machine features with a budget of 5 queries. The error bars denote standard deviations. *all-features* performs best with f1-score of 0.88, while

machine-only performs worst with f1-score around 0.815. *entropy-selection* performs only slightly worse than *all-features* from the first query, with an f1-score around 0.87. This makes sense, as the dataset was generated so that exactly 1 human feature is both uncertain and impactful for each instance, and as such, is important to query. *feature-selection* has a worse starting f1-score, around 0.83, and is unable to match the performance of *all-features* within the 5 queries. It only achieves f1-score of around 0.86 by the end. This suggests that our proposed method works to identify features that are both impactful for predictive performance, and can also be only be imperfectly predicted from the machine features.

C.3 Additional Qualitative Results

Sensible correlations between machine features and human feature queries in the recipe domain suggest reasons for performance improvements from individualized feature queries. Figure C.3 shows a heatmap of the probability a specific human feature (y-axis) will be queried using *entropy-selection* given that a specific machine feature (x-axis) is observed for the instance (computed on the test set for 1 randomly chosen random restart). Strong relationships between sensible ingredient pairings include "cucumber" and "rice vinegar", which makes sense because the ingredients commonly co-occur in various east Asian cuisines but "cucumber" is more widely used, and "turmeric" and "garam masala," which makes sense because again, "turmeric" can be used in various cuisines, but the 2 ingredients frequently co-occur in Indian cuisine. This suggests the *entropy-selection* queries are sensibly informed by the machine features.

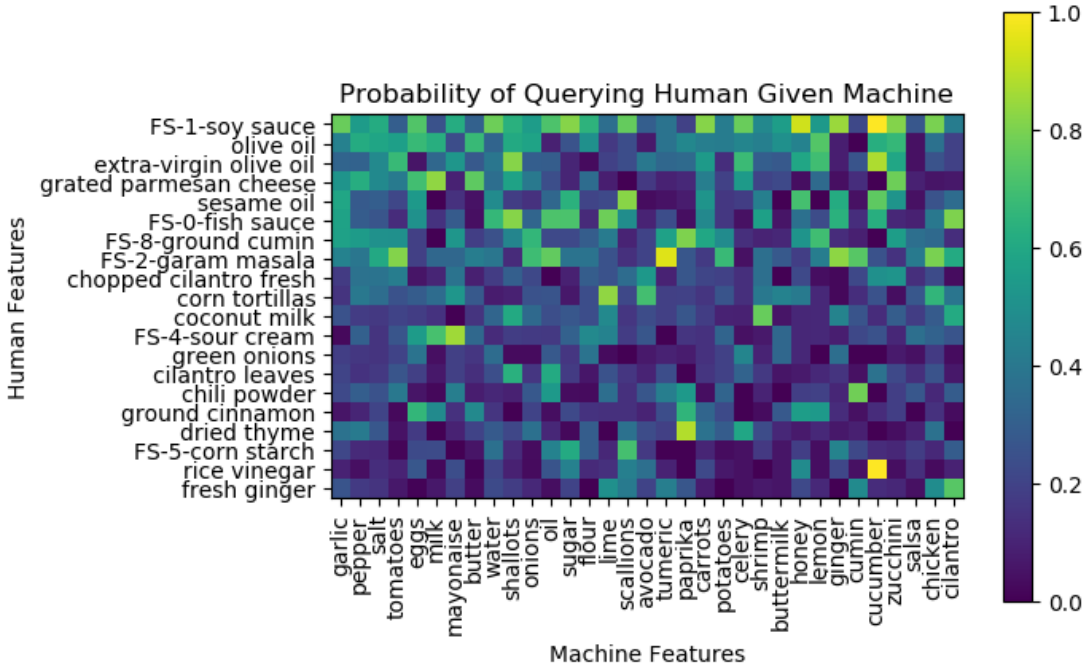


Figure C.3: Probability of querying a human feature (y-axis, top 20 sorted by # times queried) given a machine feature (x-axis) in the instance in *recipe*. Computed on test set for randomly chosen restart. "cucumber"–"rice vinegar" and "turmeric"–"garam masala" associations are sensible.

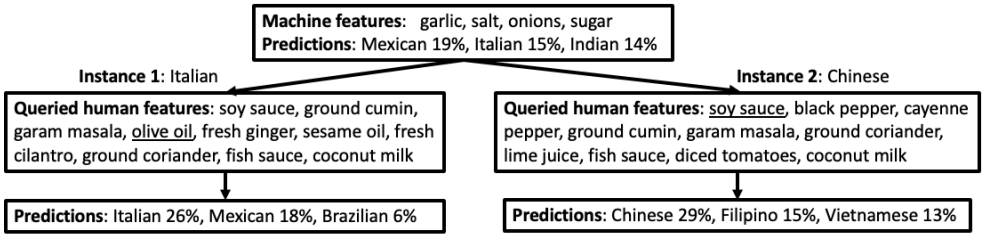


Figure C.4: Original predictions, human features queried–underlined when positive, and updated predictions for 2 test instances from a randomly chosen restart with identical machine features. Informative human features including "soy sauce" and "olive oil" help to disambiguate predictions.

An illustrative example demonstrates how different queried human feature values can cause the predictions for 2 instances with identical machine features to diverge to correct predictions. We examine in Figure C.4 how the predicted probabilities change after 10 feature queries for 2 test instances with identical machine features but different labels. While both start with Mexican cuisine as the most likely prediction, after discovering that instance 1 contains olive oil and does not contain many other distinctive ingredients including fish sauce and garam masala, the prediction correctly changes to Italian. For instance 2, the presence of soy sauce and the absence of many other features correctly changes the prediction to Chinese cuisine. Note that the queries are not identical because they depend on the values of queried human features. This example illustrates how the instance-specific human feature queries can help disambiguate the labels of otherwise similar instances.

References

[Wah] Technical report.

[2] Altendorf, E. E., Restificar, A. C., & Dietterich, T. G. (2005). Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05 (pp. 18–26). Arlington, Virginia, United States: AUAI Press.

[3] Amershi, S., Fogarty, J., Kapoor, A., & Tan, D. (2009). Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09 (pp. 247–256). New York, NY, USA: Association for Computing Machinery.

[4] Bach, F. R. (2010). Structured sparsity-inducing norms through submodular functions. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23* (pp. 118–126).: Curran Associates, Inc.

[5] Carter, G. C., Cantrell, R. A., Zarotsky, V., Haynes, V. S., Phillips, G., Alatorre, C. I., Goetz, I., Paczkowski, R., & Marangell, L. B. (2012). Comprehensive review of factors implicated in the heterogeneity of response in depression. *Depression and anxiety*, 29(4), 340–354.

[6] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721–1730).: ACM.

[7] Castro, V. M., Minnier, J., Murphy, S. N., Kohane, I., Churchill, S. E., Gainer, V., Cai, T., Hoffnagle, A. G., Dai, Y., Block, S., Weill, S. R., Nadal-Vicens, M., Pollastri, A. R., Rosenquist, J. N., Goryachev, S., Ongur, D., Sklar, P., Perlis, R. H., Smoller, J. W., , Smoller, J. W., Perlis, R. H., Lee, P. H., Castro, V. M., Hoffnagle, A. G.,

Sklar, P., Stahl, E. A., Purcell, S. M., Ruderfer, D. M., Charney, A. W., Roussos, P., Pato, C., Pato, M., Medeiros, H., Sobel, J., Craddock, N., Jones, I., Forty, L., DiFlorio, A., Green, E., Jones, L., Dunjewski, K., Landén, M., Hultman, C., Juréus, A., Bergen, S., Svantesson, O., McCarroll, S., Moran, J., Smoller, J. W., Chambert, K., & Belliveau, R. A. (2015). Validation of electronic health record phenotyping of bipolar disorder cases and controls. *American Journal of Psychiatry*, 172(4), 363–372. PMID: 25827034.

[8] Chaloner, K. & Verdinelli, I. (1995). Bayesian experimental design: A review. *Statist. Sci.*, 10(3), 273–304.

[9] Chang, J. C., Amershi, S., & Kamar, E. (2017). Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (pp. 2334–2346).

[10] Chen, J., Song, L., Wainwright, M., & Jordan, M. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning* (pp. 883–892).: PMLR.

[11] Chen, V., Plumb, G., Topin, N., & Talwalkar, A. (2021). Simulated user studies for explanation evaluation. In *eXplainable AI approaches for debugging and diagnosis.*

[12] Cheng, J. & Bernstein, M. S. (2015a). Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15 (pp. 600–611). New York, NY, USA: ACM.

[13] Cheng, J. & Bernstein, M. S. (2015b). Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing* (pp. 600–611).

[14] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4299–4307).: Curran Associates, Inc.

[15] Chu, W., Keerthi, S. S., & Ong, C. J. (2004). Bayesian support vector regression using a unified loss function. *IEEE Transactions on Neural Networks*, 15(1), 29–44.

[16] Dheeru, D. & Karra Taniskidou, E. (2017). UCI machine learning repository.

[17] Dormann, C. F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., Marquéz, J. R. G., Gruber, B., Lafourcade, B., Leitão, P. J., Münkemüller, T., McClean, C., Osborne, P. E., Reineking, B., Schröder, B., Skidmore, A. K., Zurell, D., & Lautenbach, S. (2013). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 36(1), 27–46.

[18] Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv*.

[19] Druck, G., Mann, G., & McCallum, A. (2008). Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08 (pp. 595–602). New York, NY, USA: Association for Computing Machinery.

[20] Freitas, A. A. (2014). Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1), 1–10.

[21] Green, B. & Chen, Y. (2019). Disparate interactions: An algorithm-in-the-loop analysis of fairness in risk assessments. In *Proceedings of the ACM Conference on Fairness, Accountability and Transparency*, FAT* '07.

[22] Hall, D. L. & Jordan, J. M. (2010). *Human-centered information fusion*. Artech House.

[23] Hall, D. L., McNeese, M., Llinas, J., & Mullen, T. (2008). A framework for dynamic hard/soft fusion. In *2008 11th International Conference on Information Fusion* (pp. 1–8).: IEEE.

[24] Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum*, 9(1), 926.

[25] Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

[26] Hristov, Y., Lascarides, A., & Ramamoorthy, S. (2018). Interpretable latent spaces for learning from demonstration. *CoRR*, abs/1807.06583.

[27] Hu, Y., Boyd-Graber, J., Satinoff, B., & Smith, A. (2014). Interactive topic modeling. *Machine Learning*, 95(3), 423–469.

[28] Jacobs, M., He, J., F. Pradier, M., Lam, B., Ahn, A. C., McCoy, T. H., Perlis, R. H., Doshi-Velez, F., & Gajos, K. Z. (2021). Designing ai for trust and collaboration in time-constrained medical decisions: A sociotechnical lens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* New York, NY, USA: Association for Computing Machinery.

[29] Jethani, N., Sudarshan, M., Aphinyanaphongs, Y., & Ranganath, R. (2021). Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics* (pp. 1459–1467).: PMLR.

[30] Jordan, M. I. & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2), 181–214.

[31] Joshi, S., Parbhoo, S., & Doshi-Velez, F. (2021). Pre-emptive learning-to-defer for sequential medical decision-making under uncertainty. *arXiv preprint arXiv:2109.06312*.

[32] Kim, B., Rudin, C., & Shah, J. A. (2014). The bayesian case model: A generative approach for case-based reasoning and prototype classification. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 1952–1960).: Curran Associates, Inc.

[33] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., & Sayres, R. (2017). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav).

[34] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., & sayres, R. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research* (pp. 2668–2677). Stockholmsmässan, Stockholm Sweden: PMLR.

[35] Koh, P. W., Nguyen, T., Tang Yew Siang, Mussmann, S., Emma, P., Kim, B., & Liang, P. (2020). Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*.

[36] Kosorukoff, A. (2001). Human based genetic algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 5.

[37] Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S., & Doshi-Velez, F. (2019). Human evaluation of models built for interpretability. In *Proceedings of the Seventh AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '19 New York, NY, USA: AAAI Press.

[38] Lage, I., McCoy Jr, T. H., Perlis, R. H., & Doshi-Velez, F. (2022). Efficiently identifying individuals at high risk for treatment resistance in major depressive disorder using electronic health records. *Journal of Affective Disorders*, 306, 254–259.

[39] Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016a). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (pp. 1675–1684). New York, NY, USA: ACM.

[40] Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016b). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1675–1684).

[41] Lavrac, N. (1999). Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(1), 3 – 23. Data Mining Techniques and Applications in Medicine.

[42] Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.*, 9(3), 1350–1371.

[43] Lipton, Z. C. (2016a). The mythos of model interpretability. *CoRR*, abs/1606.03490.

[44] Lipton, Z. C. (2016b). The mythos of model interpretability. *CoRR*, abs/1606.03490.

[45] Little, G., Chilton, L. B., Goldman, M., & Miller, R. C. (2010). Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10 (pp. 57–66). New York, NY, USA: ACM.

[46] Lund, J., Cook, C., Seppi, K., & Boyd-Graber, J. (2017). Tandem anchoring: a multiword anchor approach for interactive topic modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers) (pp. 896–905). Vancouver, Canada: Association for Computational Linguistics.

[47] Lund, J., Cowley, S., Fearn, W., Hales, E., & Seppi, K. (2018). Labeled anchors and a scalable, transparent, and interactive classifier. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 824–829). Brussels, Belgium: Association for Computational Linguistics.

[48] Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., Liston, D. E., Low, D. K.-W., Newman, S.-F., Kim, J., & Lee, S.-I. (2017). Explainable machine learning predictions to help anesthesiologists prevent hypoxemia during surgery. *bioRxiv*.

[49] Ma, Y., Garnett, R., & Schneider, J. G. (2012). Submodularity in batch active learning and survey problems on gaussian random fields. *CoRR*, abs/1209.3694.

[50] Madras, D., Pitassi, T., & Zemel, R. (2018). Predict responsibly: improving fairness and accuracy by learning to defer. *Advances in Neural Information Processing Systems*, 31.

[51] Masood, M. A. & Doshi-Velez, F. (2018). A particle-based variational approach to bayesian non-negative matrix factorization. *arXiv*.

[52] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267, 1–38.

[53] Mozannar, H. & Sontag, D. (2020). Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning* (pp. 7076–7087).: PMLR.

[54] Narayanan, M., Emily, Chen, He, J., Kim, B., Gershman, S., & Doshi-Velez, F. (2018). How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation. *ArXiv e-prints*.

[55] Narayanaswamy, S., Paige, T. B., van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., & Torr, P. (2017). Learning disentangled representations with semi-supervised deep generative models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 5925–5935). Curran Associates, Inc.

[56] Parbhoo, S., Bogojeska, J., Zazzi, M., Roth, V., & Doshi-Velez, F. (2017). Combining kernel and model based learning for hiv therapy selection. *AMIA Summits on Translational Science Proceedings*, 2017, 239.

[57] Parikh, D. & Grauman, K. (2011). Interactively building a discriminative vocabulary of nameable attributes. In *CVPR 2011* (pp. 1681–1688).

[58] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011a). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.

[59] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011b). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[60] Poote, A. E., French, D. P., Dale, J., & Powell, J. (2014). A study of automated self-assessment in a primary care student health centre setting. *Journal of telemedicine and telecare*, 20(3), 123–127.

[61] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., & Wallach, H. M. (2018a). Manipulating and measuring model interpretability. *CoRR*, abs/1802.07810.

[62] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., & Wallach, H. M. (2018b). Manipulating and measuring model interpretability. *CoRR*, abs/1802.07810.

[63] Pradier, M. F., McCoy Jr, T. H., Hughes, M., Perlis, R. H., & Doshi-Velez, F. (2020). Predicting treatment dropout after antidepressant initiation. *Translational Psychiatry*, 10(1), 60.

[64] Pradier, M. F., Zazo, J., Parbhoo, S., Perlis, R. H., Zazzi, M., & Doshi-Velez, F. (2021). Preferential mixture-of-experts: Interpretable models that rely on human expertise as much as possible. *AMIA Summits on Translational Science Proceedings*, 2021, 525.

[65] Quost, B. (2021). Decision-making from partial instances by active feature querying. In *International Symposium on Imprecise Probability: Theories and Applications* (pp. 264–272).: PMLR.

[66] Raghavan, H., Madani, O., & Jones, R. (2006). Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7(61), 1655–1686.

[67] Raghu, M., Blumer, K., Corrado, G., Kleinberg, J., Obermeyer, Z., & Mullainathan, S. (2019). The algorithmic automation problem: Prediction, triage, and human effort. *arXiv preprint arXiv:1903.12220*.

[68] Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT Press.

[69] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (pp. 1135–1144). New York, NY, USA: ACM.

[70] Ritchie, M. D., Denny, J. C., Crawford, D. C., Ramirez, A. H., Weiner, J. B., Pulley, J. M., Basford, M. A., Brown-Gentry, K., Balser, J. R., Masys, D. R., Haines, J. L., & Roden, D. M. (2010). Robust replication of genotype-phenotype associations across multiple diseases in an electronic medical record. *The American Journal of Human Genetics*, 86(4), 560 – 572.

[71] Rokach, L. & Maimon, O. (2014). *Introduction to Decision Trees*, chapter Chapter 1, (pp. 1–16). WORLD SCIENTIFIC, 2nd edition.

[72] Ross, A., Lage, I., & Doshi-Velez, F. (2017). The neural lasso: Local linear sparsity for interpretable explanations. In *Workshop on Transparent and Interpretable Machine Learning in Safety Critical Environments, 31st Conference on Neural Information Processing Systems*. https://goo.gl/TwRhXo.

[73] Ross, A., Pan, W., & Doshi-Velez, F. (2018). Learning qualitatively diverse and interpretable rules for classification. In *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*.

[74] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206–215.

[75] Ruotsalo, T. & Lipsanen, A. (2018). Interactive symptom elicitation for diagnostic information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1301–1304).

[76] Saar-Tsechansky, M., Melville, P., & Provost, F. (2009). Active feature-value acquisition. *Management Science*, 55(4), 664–684.

[77] Sakata, Y., Baba, Y., & Kashima, H. (2019). Crownn: Human-in-the-loop network with crowd-generated inputs. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 7555–7559).: IEEE.

[78] Sankaranarayanan, K. & Dhurandhar, A. (2013). Intelligently querying incomplete instances for improving classification performance. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 2169–2178).

[79] Settles, B. (2009). Active learning literature survey.

[80] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 2951–2959).: Curran Associates, Inc.

[81] Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2009). *Gaussian Process Bandits without Regret: An Experimental Design Approach*. Technical Report arXiv:0912.3995. Comments: 17 pages, 5 figures.

[82] Takahama, R., Baba, Y., Shimizu, N., Fujita, S., & Kashima, H. (2018). Adaflock: Adaptive feature discovery for human-in-the-loop predictive modeling. In *AAAI*.

[83] Tamuz, O., Liu, C., Belongie, S. J., Shamir, O., & Kalai, A. T. (2011). Adaptively learning the crowd kernel. *CoRR*, abs/1105.1033.

[84] Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. *Data classification: Algorithms and applications*, (pp.37).

[85] Tibshirani, R. (1996a). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, (pp. 267–288).

[86] Tibshirani, R. (1996b). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, (pp. 267–288).

[87] Townsend, L., Walkup, J. T., Crystal, S., & Olfson, M. (2012). A systematic review of validated methods for identifying depression using administrative data. *Pharmacoepidemiology and Drug Safety*, 21(S1), 163–173.

[88] Tran-Thanh, L., Huynh, T. D., Rosenfeld, A., Ramchurn, S., & Jennings, N. R. (2014). Budgetfix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees.

[89] Ustun, B. & Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3), 349–391.

[90] Ustun, B. & Rudin, C. (2017). Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17 (pp. 1125–1134). New York, NY, USA: ACM.

[91] Wang, J., Oh, J., Wang, H., & Wiens, J. (2018). Learning credible models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2417–2426).

[92] Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., & MacNeille, P. (2017). A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1), 2357–2393.

[93] Wilder, B., Horvitz, E., & Kamar, E. (2020). Learning to complement humans. *arXiv preprint arXiv:2005.00582*.

[94] Wilson, A. G., Dann, C., Lucas, C., & Xing, E. P. (2015). The human kernel. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 2854–2862).: Curran Associates, Inc.

[95] Wirth, C., Akrour, R., Neumann, G., & Fürnkranz, J. (2017). A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136), 1–46.

[96] Wu, M., Hughes, M. C., Parbhoo, S., Zazzi, M., Roth, V., & Doshi-Velez, F. (2018). Beyond Sparsity: Tree Regularization of Deep Models for Interpretability. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

[97] Yoon, J., Jordon, J., & van der Schaar, M. (2018). Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*.

[98] Zhu, X., Lafferty, J., & Ghahramani, Z. (2003a). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining* (pp. 58–65).

[99] Zhu, X., Lafferty, J., & Ghahramani, Z. (2003b). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3.

[100] Zou, J., Chaudhuri, K., & Kalai, A. (2015). Crowdsourcing feature discovery via adaptively chosen comparisons. In *Third AAAI Conference on Human Computation and Crowdsourcing*.

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, "Science Experiment 02", was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD thesis with this look and feel has been released under the permissive MIT (X11) license, and can be found online at github.com/suchow/Dissertate or from its author, Jordan Suchow, at suchow@post.harvard.edu.